# PROYECTO DE IMPLEMENTACIÓN Y MONITORIZACIÓN DE UN CLÚSTER DE SERVIDORES WEB LINUX CON BALANCEO DE CARGA

**Autor:** José Cristian Cañaveras Vélez

# ÍNDICE

# 1. Introducción

Este documento elaborado para el proyecto del ciclo Superior de Administración de Sistemas Informáticos en Red del IES Gregorio Prieto. En el proyecto se realiza una implementación de un cluster de servidores Web con balanceo de carga para obtener alta disponibilidad.

Gracias al clúster de servidores podemos tener varios servidores unidos en un "clúster" o varios, permitiendo que si uno de ellos falla el servicio siga funcionando con normalidad y ofreciendo los servicios web.

Para evitar la sobrecarga de los servidores (caídas, debido a intentar resolver más peticiones que las que pueden resolver) se implementara un sistema de balanceo de carga, cuya función es enviar las peticiones web realizadas por los clientes a cada grupo de servidores (clúster) y así repartir la carga entre los servidores disponibles evitando la sobrecarga.

Para tener controlado el conjunto de servidores, balanceadores de carga, etc. se dispondrá de un sistema de monitorización web intuitivo que permite observar el estado de los servidores y en caso de fallo se enviara un correo electrónico con el fallo y se podrá visualizar en el panel de control denominado "**clusterCP**".

# 2. Objetivos y aspectos que se pretenden abordar

En este proyecto se pretenden abordar los siguientes objetivos.

- Reducir la sobrecarga de los servidores, repartiéndola entre los servidores disponibles.
- Diseñar una infraestructura web tolerante a fallos.
- Almacenamiento de los archivos web permitiendo disminuir el tiempo de acceso a la web, ya que los archivos están disponibles en cada servidor y así evitar la pérdida de datos si un servidor falla.
- Disponer de un sistema de monitorización sencillo para identificar posibles incidencias.
- Monitorización 24/7/365 gracias a las notificaciones por correo electrónico.

A la finalización de este proyecto dispondremos de una infraestructura web tolerante a fallos en los servidores y obtendremos un mayor rendimiento an cada servidor al no usar todos los recursos.

## 3. Recursos y Medios que se utilizarán.

La elaboración de este proyecto se ha realizado de forma virtualizada, se ha utilizado un Equipo Portátil con las siguientes características:

- Procesador: Intel Core i7-4710HQ
- Memoria RAM: 16 GB
- Disco duro 1: SSD M-SATA 250GB
- Disco duro 2: SATA 1TB
- Disco duro 3: SATA 750GB
- S.O: Windows 7 Professional x64

La virtualización se ha realizado usando el software VirtualBox y para la emulación de red se ha usado el software GNS3.

Los sistemas operativos usados para la realización del proyecto son:

- CentOS 7 x64
- Ubuntu Server 14.04.2 x64
- Debian 8.0 x64
- ElemetaryOS freya  x64

## 4. Presupuesto

Si la infraestructura la vamos a poner en producción (real) deberemos invertir dinero en una infraestructura de servidores, red, etc.

Debemos tener en cuenta las ganancias que generaremos con nuestra infraestructura, ya que si la infraestructura web debe estar operativa 24 horas 7 días a la semana y ofrecer seguridad deberemos invertir.

Un ejemplo es una entidad financiera en la que la plataforma web y la "intranet", deberá estar operativa en todo momento, por lo que es necesario invertir en infraestructura y alta disponibilidad ya que si no perderá dinero.

A continuación se especifica un presupuesto estimado de la inversión necesaria para crear la infraestructura de servidores y Red.

### Servidores

#### Clústers

Para los servidores del clúster se han elegido servidores DELL PowerEdge R220 con 16 GB de memoria RAM, 500GB de disco SSD para el sistema operativo y un disco de 1TB SSD para los datos de Apache y obtener la máxima rapidez.

**Precio:**

- Servidor → 667 € x 4 servidores = 2668€

  http://www.marketdirecto.com/dell-poweredge-r220-742775-p.htm

- Memoria RAM (módulo de 8GB) → 4 servidores x 2 modulos/servidor x 58€/modulo = 464€

  http://www.pccomponentes.com/kingston_valueram_ddr3_1600_pc3_12800_1x8gb_cl11.html

- Procesador (Intel Xeon E3-1220 V3 3.10 GHz ) → 4 servidores x 224 € = 896€

  http://www.pccomponentes.com/intel_xeon_e3_1220_v3_3_10_ghz_box.html

- Disco Duro SSD 500GB →4 servidores x 189 € = 756€

  http://www.pccomponentes.com/samsung_850_evo_ssd_series_500gb__sata3.html

- Disco Duro SSD 1TB →4 servidores x 386 € = 1544€

  http://www.pccomponentes.com/samsung_850_evo_ssd_series_1tb__sata3.html

- Sistema Operativo: CentOS = 0€

**TOTAL = 2668€ + 464 € + 896 € + 756 € + 1544€ = 6328 €**

## Balanceadores

Para los servidores de balanceo de carga se han elegido servidores DELL PowerEdge T110 II con 8GB de memoria RAM, 250GB SSD de disco duro.

**Precio:**

- Servidor → 380 € x 2 servidores = 760€

  http://www.marketdirecto.com/dell-poweredge-r220-742775-p.htm

- Memoria RAM (módulo de 8GB) → 2 servidores x 1 modulos/servidor x 42,84 €/modulo = 85,68€

  http://www.pccomponentes.com/kingston_valueram_ddr3_1600_pc3_12800_1x8gb_cl11.html

- Procesador (Intel Xeon E3-1240 V2 3.40 GHz ) → 2 servidores x 170 € = 340€

  http://configure.euro.dell.com/dellstore/config.aspx?oc=svt110ii&model_id=poweredge-t110-2&c=es&l=es&s=bsd&cs=esbsdt1

- Disco Duro SSD 250GB →2 servidores x 107 € = 214€

  http://www.pccomponentes.com/samsung_850_evo_ssd_series_250gb__sata3.html

- Sistema Operativo: Ubuntu Server = 0€

**TOTAL = 760€ + 85,68 € + 340 € + 214 € = 1399 €**

## Red

Para la red instalaremos switch Cisco Catalyst 2950 de 24 puertos GigaByte Ethernet, para el enrutamiento con el exterior se instalara un router Cisco

- Switch (Cisco Catalyst 2950 24 puertos) → 7 x 256 € = 1792 €

  http://www.amazon.com/Cisco-WS-C2950T-24-Catalyst-2950-Switch/dp/B00007MD88

- Router (Cisco 3925 Series ) → 1 x 2877,25 € = 2877,25 €

  http://www.amazon.com/Cisco-Router-CISCO3925-K9-cisco/dp/B008TSLYC8

**TOTAL = 1792€ + 2877,25 € = 4669,25 €**

**TOTAL PRESUPUESTO (ESTIMADO): 6328 € + 1399 € + 4669,25 € = 12396 €**

**NOTA: Este presupuesto es estimado, ya que hay que añadirle la infraestructura donde se albergaran los servidores, cableado, acceso a Internet, instalación y configuración de servidores, también deberemos tener en cuenta la instalaci'on de sistemas UPS, etc.**

## 5. Despliegue de la Red y configuración inicial

Nuestro despliegue de servidores consta de los siguientes SS.OO:

- Clúster (A y B): CentOS 7 x64 (1GB RAM, 20GB HD)
- Balanceador de Carga (LB1 y LB2): Ubuntu Server 14.04.2 x64 (1GB RAM, 20GB HD)
- Monitor (Sistema de monitorización del clúster): Debian 8.0 x64 (1GB RAM, 30GB HD)
- Cliente (PC de un usuario cliente): ElemetaryOS freya  x64 (1GB RAM, 20GB HD)

NOTA: Más adelante se indicaran los discos duros extras para la replicación de datos a través de GlusterFS.

La configuración de red será la siguiente:

**CLUSTER A (**IP flotante: **192.168.50.254/24)**

- nodoa1
  - enp0s8 (eth1): 192.168.50.10/24
  - enp0s9 (eth2): 172.16.50.10/16
  - enp0s10 (eth3): 172.17.50.10/16
- nodoa2
  - enp0s8 (eth1): 192.168.50.11/24
  - enp0s9 (eth2): 172.16.50.11/16
  - enp0s10 (eth3): 172.17.50.11/16

**CLUSTER B (**IP flotante: **192.168.50.253/24)**

- nodob1
  - enp0s8 (eth1): 192.168.50.15/24
  - enp0s9 (eth2): 172.16.50.15/16
  - enp0s10 (eth3): 172.17.50.15/16
- nodob2
  - enp0s8 (eth1): 192.168.50.16/24
  - enp0s9 (eth2): 172.16.50.16/16
  - enp0s10 (eth3): 172.17.50.16/16

**LOAD BALANCER (**IP flotante: **192.168.20.254/24)**

- lb1
  - eth1: 192.168.20.10/24
  - eth2: 192.168.50.200/24
- lb2
  - eth1: 192.168.20.11/24
  - eth2: 192.168.50.201/24

**MONITOR**

- mon1
  - eth1: 192.168.50.190/24

**DNS**

- dns01
  - eth1: 192.168.20.200/24

En el siguiente gráfico se visualiza la infraestructura de red y servidores.

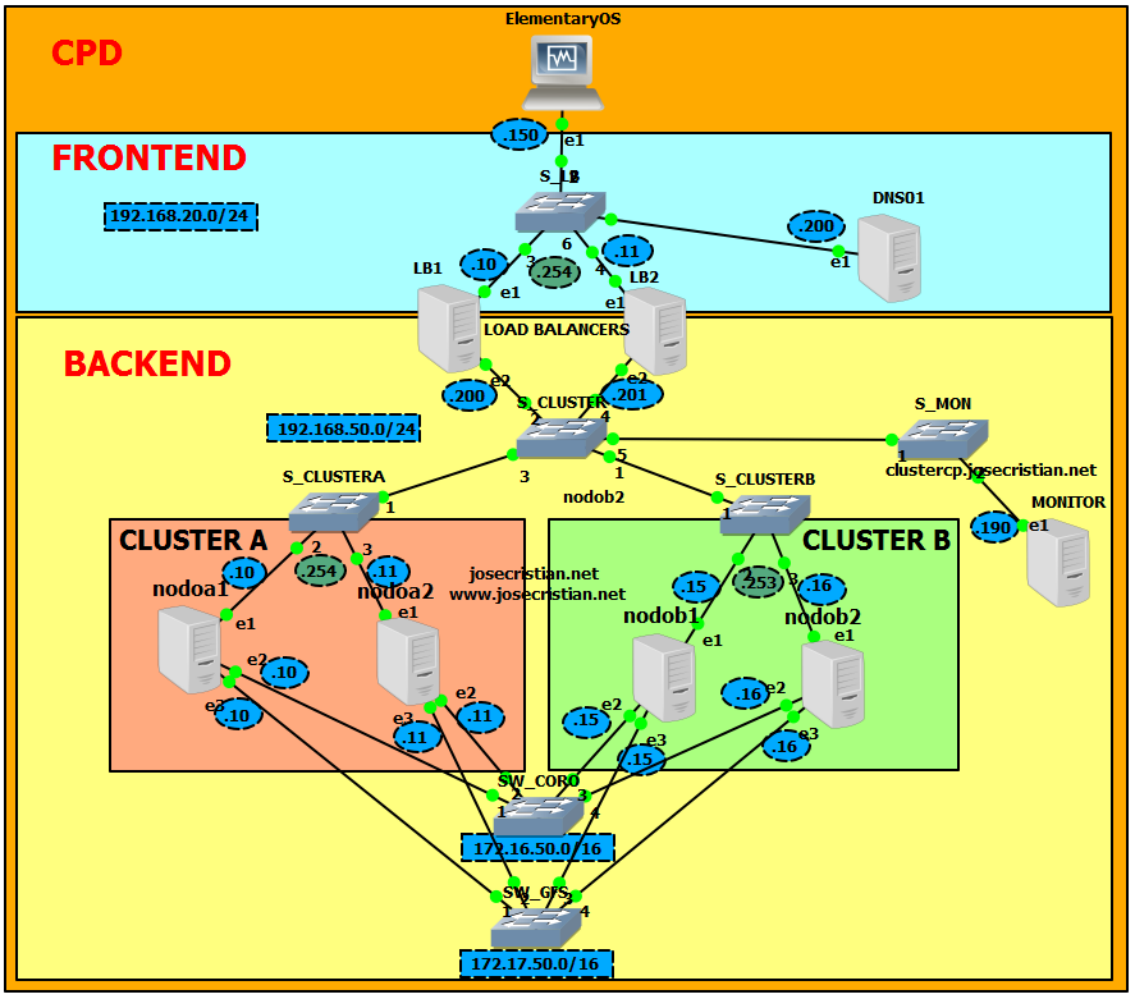


***Imagen 5.1. Esquema del proyecto.***

## 5.1. Instalación del S.O

Para evitar alargar el documento documentaré la instalación de un S.O del CLUSTER, LOAD BALANCER y MONITOR (clusterCP), ya que los demás servidores se instalan de la misma manera a diferencia de la configuración de red y el nombre del equipo que es el correspondiente a cada servidor.

**CLUSTER**

El S.O para los servidores del clúster será CentOS 7 x64 con 1Gb de RAM y 20 GB de HD. Esta instalación será para el servidor nodoa1

Para ello descargamos la imange ISO de la web oficial de CentOS (http://isoredirect.centos.org/centos/7/isos/x86_64/CentOS-7-x86_64-DVD-1503-01.iso)

Una vez descargada arrancamos el servidor con la Imagen ISO.

Elegimos el idioma de instalación, Español.

Realizaremos una instalación con una GUI mínima, es decir, interfaz consola.

Establecemos el nombre del equipo que será **nodoa1** (la configuración de red la realizaremos más adelante)

Ahora mientras se instala el S.O configuraremos la contraseña para el usuario root y para el usuario administrador.

La contraseña para **root** será **inves** (por comodidad para la realización del proyecto)

La contraseña para el usuario **jccvelez** será **claveinves** (por comodidad para la realización del proyecto)

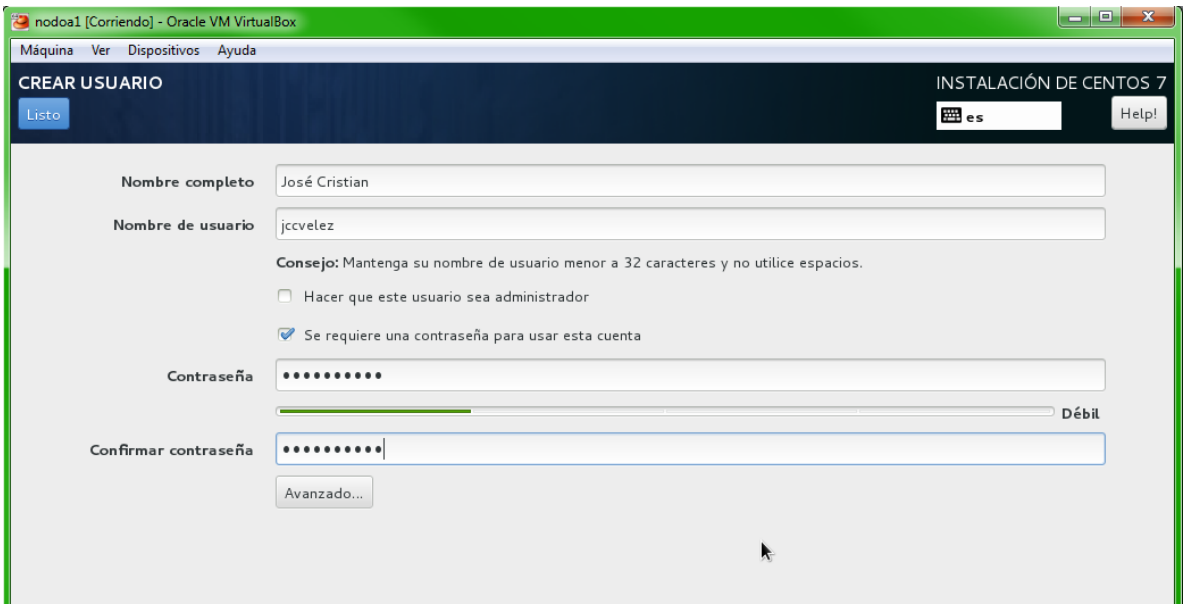


***Imagen 5.1.1. Usuario del servidor.***

Una vez terminada la instalación configuraremos las tarjetas de red, para ello configuraremos las interfaces editando el archivo de la interfaz ubicado en la ruta "**/etc/sysconfig/network-scripts/ifcfg-enp0sX**" (donde X es el número de interfaz)

Antes instalaremos la utilidad **ifconfig** ya que en CentOS 7 en la instalación mínima no viene instalada.

```
#yum isntall net-tools
```

### ifcfg-enp0s8 (192.168.50.10/24)

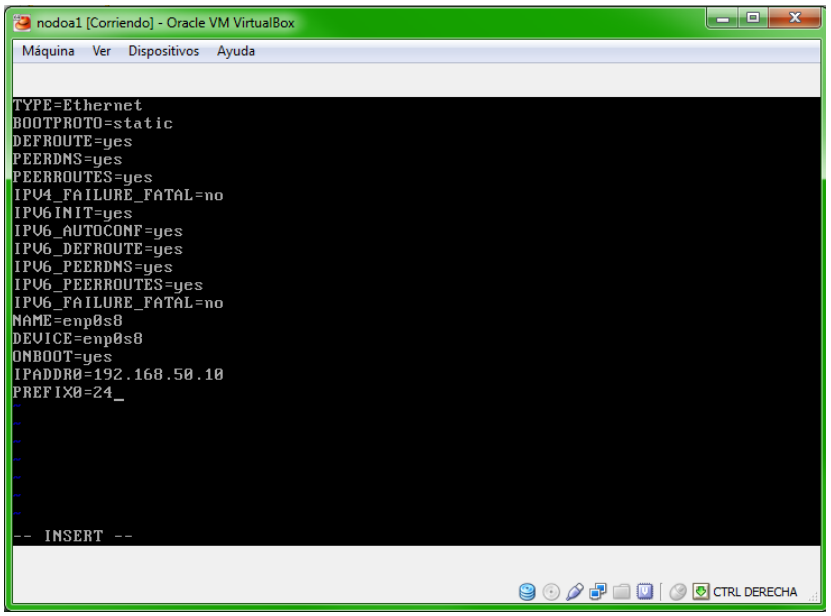


***Imagen 5.1.2. Configuracion de Red.***

### ifcfg-enp0s9 (172.16.50.10/16)

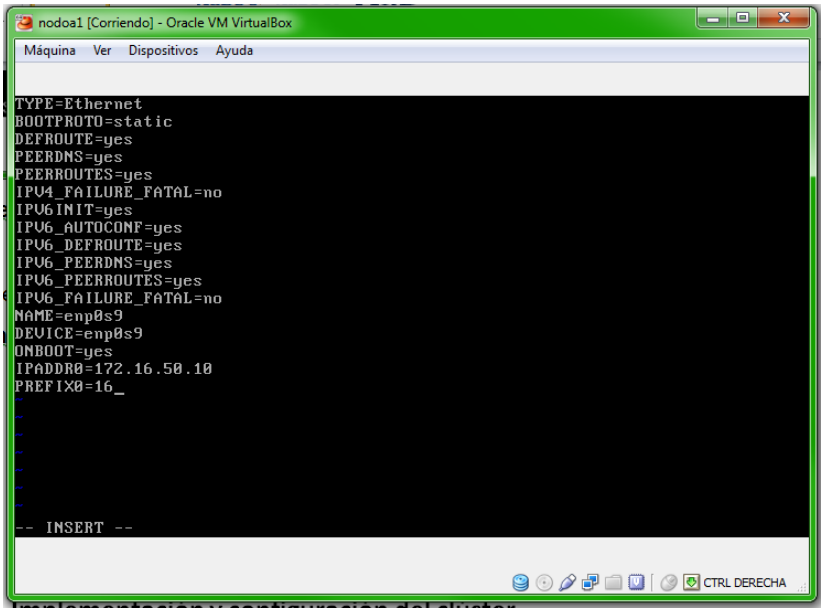


***Imagen 5.1.3. Configuracion de Red.***

**ifcfg-enp0s9 (172.17.50.10/16)**

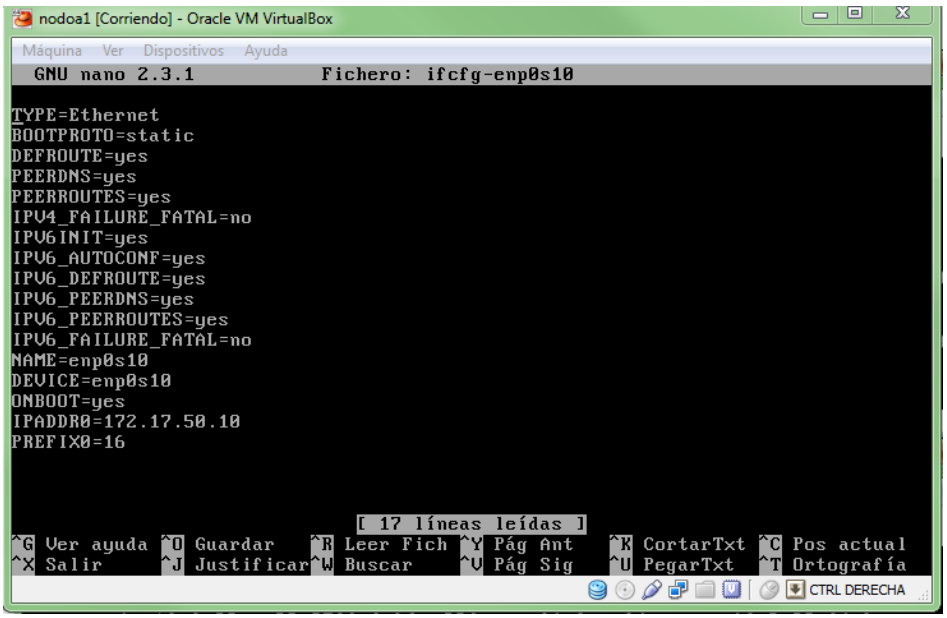


***Imagen 5.1.4. Configuracion de Red.***

Una vez configuradas las tarjetas de red las activaremos.

```
#ifdown enp0s8
#ifup enp0s8
#ifdown enp0s9
#ifup enp0s9
```

Una vez terminada la instalación actualizaremos el sistema.

```
# yum update
```

**Realizaremos el mismo procedimiento con el resto de nodos que formaran parte de cada clúster.**

## LOAD BALANCER

El S.O para los servidores encargados del **balanceo de carga** será Ubuntu Server 14.04.2 x64 con 1Gb de RAM y 20 GB de HD. Esta instalación será para el servidor LB1.

Para ello descargamos la imagen ISO de la web oficial de CentOS (http://releases.ubuntu.com/14.04.2/ubuntu-14.04.2-server-amd64.iso)

El usuario creado será **jccvelez** con la contraseña **claveinves (**por comodidad**).**

Una vez terminada la instalación configuraremos las tarjetas de red **eth1** y **eth2** con la configuración de red indicada anteriormente.

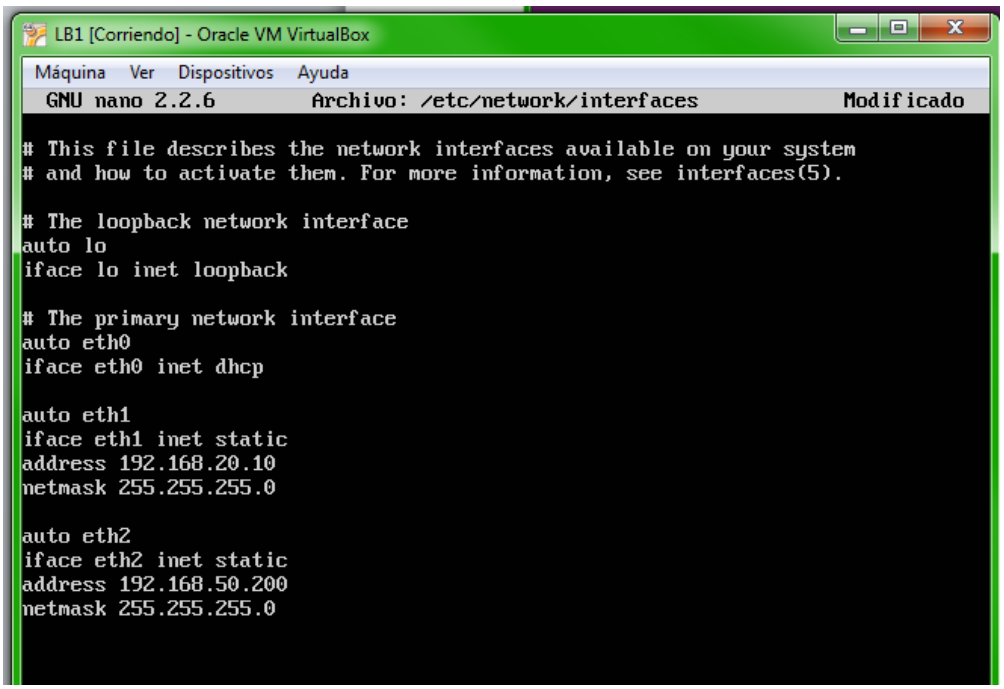


***Imagen 5.1.5. Configuracion de Red.***

Ahora actualizaremos el sistema

```
$sudo apt-get update && apt-get upgrade
```

**Realizaremos el mismo procedimiento con el balanceador LB2.**

**MONITOR**

Este equipo usara el S.O Debian 8.0 Jessie, para ello descargaremos la imagen ISO de la web oficial (http://cdimage.debian.org/debian-cd/8.0.0/amd64/iso-cd/) y la instalaremos.

Crearemos el usuario **jccvelez** con contraseña **claveinves** (por comodidad).

Una vez instalado configuraremos la tarjeta de red con la dirección IP indicada anteriormente (192.168.50.190/24).


***Imagen 5.1.6. Configuracion de Red.***

Ahora actualizaremos el sistema ejecutando

```
#apt-get update && apt-get upgrade
```

Una vez instalados los sistemas tendremos la infraestructura preparada para configurar el clúster y el balanceo de carga en los siguientes puntos.

# 6. Implementación y configuración del clúster

Explicaré la configuración del **CLUSTER A** ya que el **CLUSTER B** se realiza de la misma manera.

Realizaremos dos clusters para dotar a nuestra instalación de Alta disponibilidad, para ello usaremos **corosync** cuya función es la de proporcionar la comunicación entre los nodos para informar de errores en los nodos, **pacemaker** proporciona el control de recursos en cada nodo del cluster para en caso de notificación de corosync se asigne en recurso a otro nodo disponible, finalmente usaremos **pcs** para la autenticación entre los nodos y el mecanismo de control, también permite el arranque, parada, etc. del cluster.

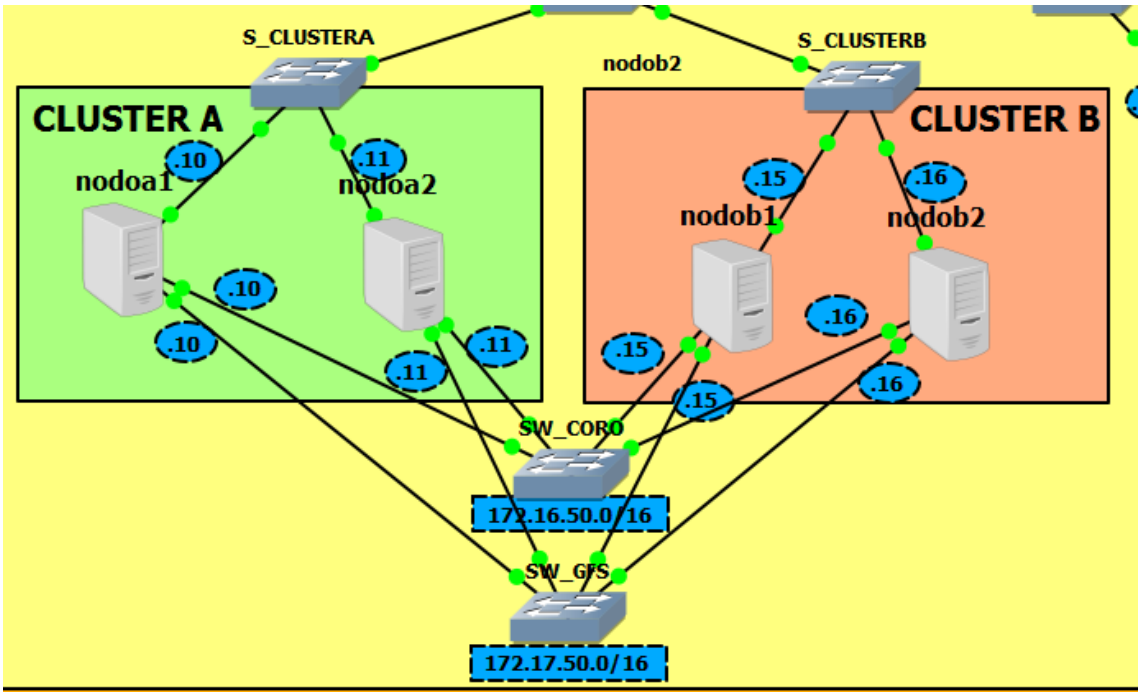


*Imagen 6.1. Esquema del clúster.*

Instalamos corosync, pacemaker y pcs en ambos nodos (nodoa1 y nodoa2).

```
#yum install corosync pacemaker pcs
```

Ahora configuraremos los nombres de los nodos en el archivo **/etc/hosts**

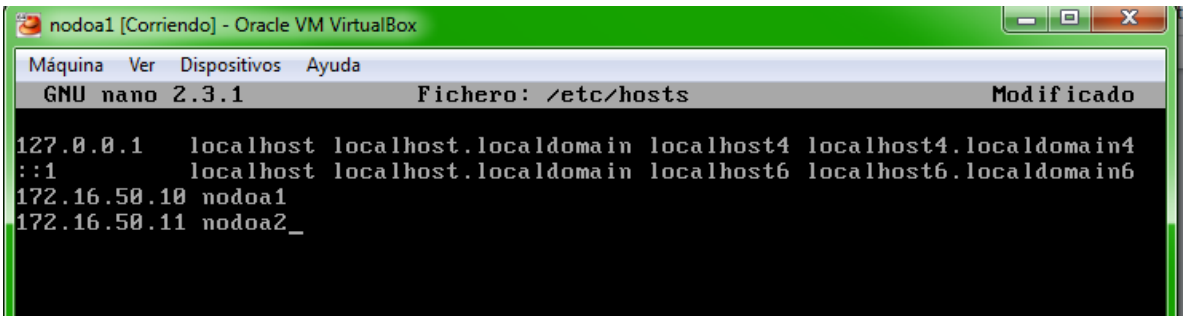


***Imagen 6.2. Archivo Hosts.***

NOTA: Si disponemos de un servidor DNS podemos centralizar los nombres de cada nodo.

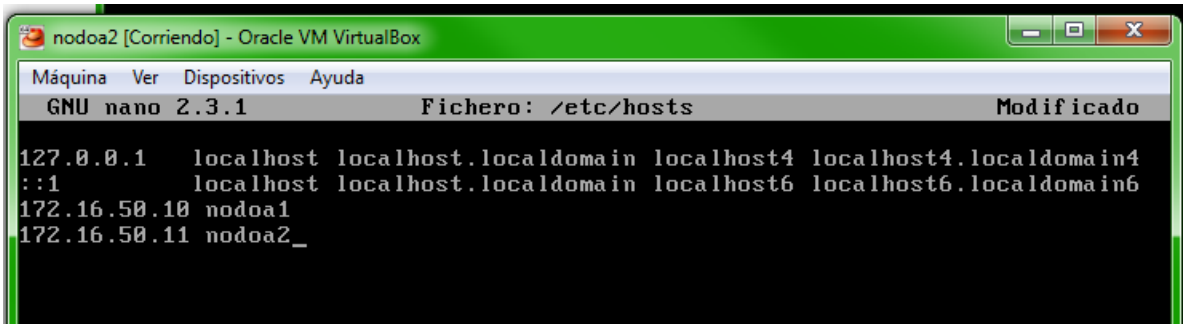


***Imagen 6.3. Archivo Hosts.***

Una vez terminada la instalación habilitaremos los servicios de los paquetes instalados en ambos nodos.

```
#systemctl start pcsd.service
#systemctl start corosync.service
#systemctl start pacemaker.service
```

NOTA: Los dos últimos servicios darán error ya que aún no están configurados.

Una vez instalado e iniciado los servicios configuraremos la contraseña (en ambos nodos) para el usuario **hacluster** que es el usuario que se crear por defecto en la instalación de **pcs**.

```
#passwd hacluster
```

La contraseña será **claveinves (**por comodidad**)**

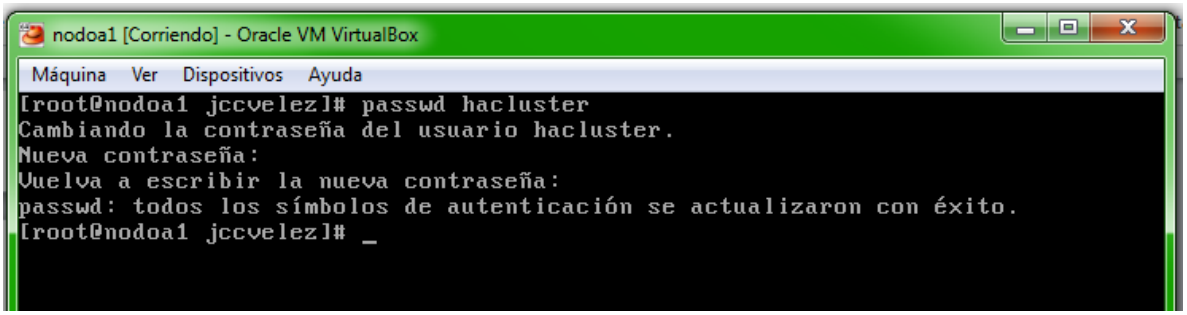


*Imagen 6.4. Contraseña del usuario del clúster.*

**NOTA: La contraseña debe de ser idéntica en ambos nodos.**

A partir de este momento trabajaremos únicamente sobre el nodo **nodoa1** ya que el nodoa2 se configurara automáticamente gracias a **pcs.**

Ahora autenticaremos los nodos del CLUSTER A que serán nodoa1 y nodoa2 y nos identificaremos con el usuario **hacluster** y la contraseña configurada.

```
#pcs cluster auth nodoa1 nodoa2
```

Ahora instalaremos el clúster con los nodos identificados anteriormente, el clúster se llamara **clustera.**

```
#pcs cluster setup --name clustera nodoa1 nodoa2
```

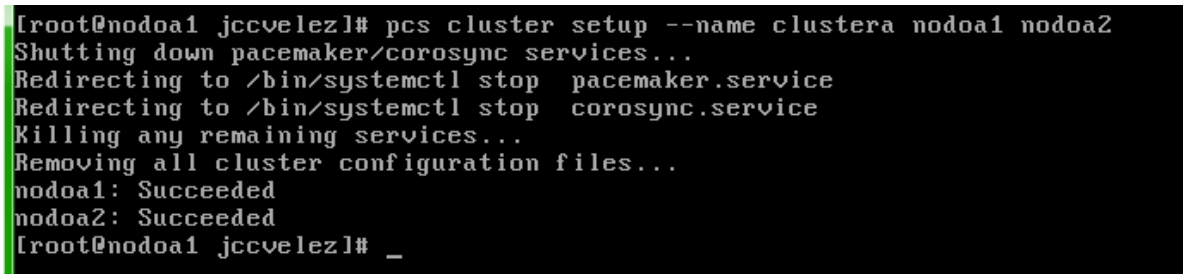


*Imagen 6.5. Instalacion de nodos y creación del clúster.*

Iniciaremos el clúster en todos los nodos del clúster **CLUSTER A** usando **pcs.**

```
#pcs cluster start --all
```

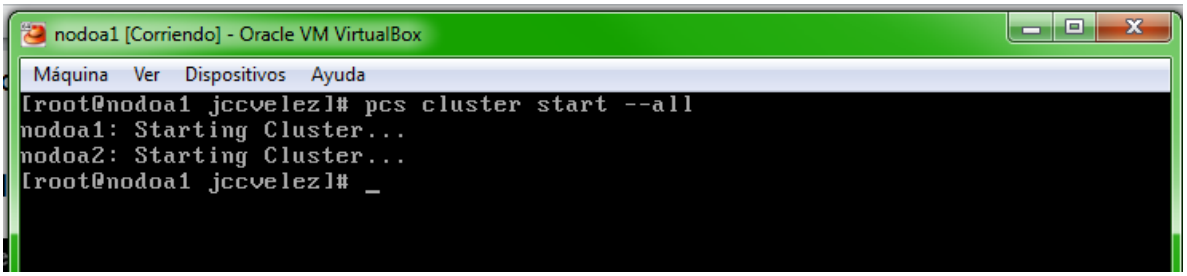


***Imagen 6.6. Inicio de los clústers.***

Ahora configuraremos una directiva para evitar que el nodo se reinicie cuando vaya mal usando la configuración **stonith**, por lo que la deshabilitaremos.

```
#pcs property set stonith-enabled=false
```

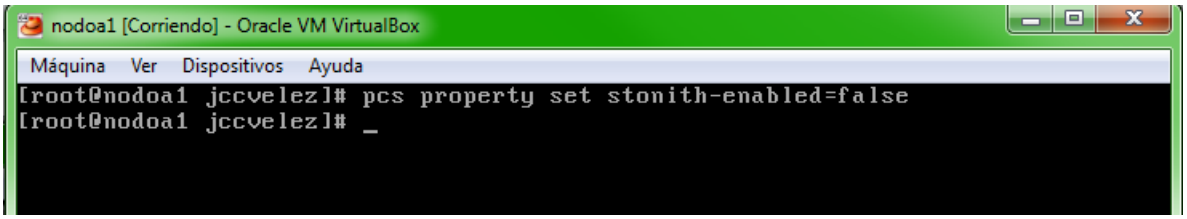


***Imagen 6.7. Evitar el reinicio en caso de error.***

Una vez configurado **stonith**, configuraremos las opciones del cluster.

Ahora ignoraremos la falta de quórum ya que se dice que un cluster tiene quórum cuando más de la mitad de los nodos que forman un cluster están online, en este caso como solo tenemos dos nodos al fallar uno de ellos habría un error por lo que ignoraremos el uso del quórum.

```
#pcs property set no-quorum-policy=ignore
```

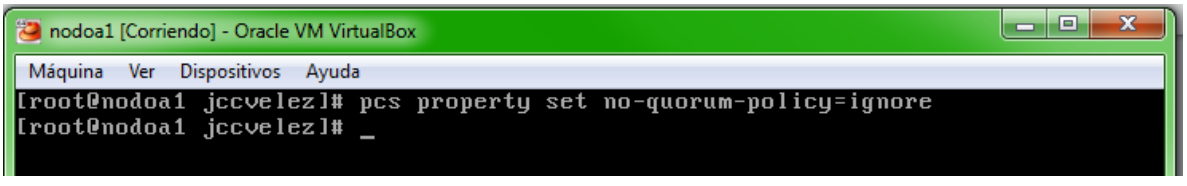


***Imagen 6.8. Ignorar falta de quorum.***

Indicaremos que el clúster se inicie desde cualquier nodo del cluster con la propiedad "**symmetric-cluster**" ("true"), ya que no queremos que se indique el nodo a arrancar, en este caso indicaríamos "false"

```
#pcs property set symmetric-cluster="true"
```

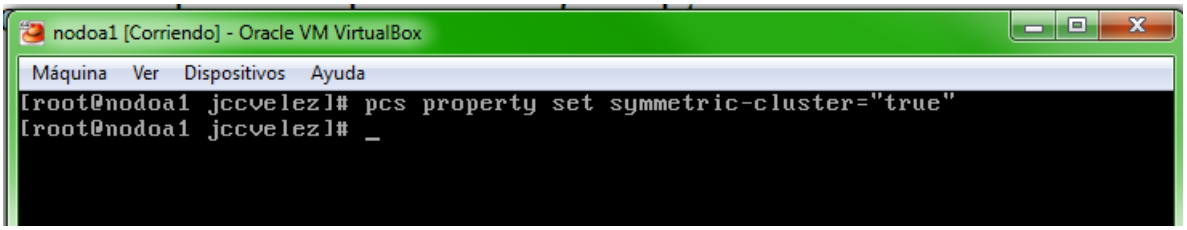


***Imagen 6.9. Inicio del clúster desde cualquier nodo.***

Una vez configurados los nodos crearemos la IP flotante (dirección IP compartida entre todos los nodos del cluster), que será la **192.168.50.254/24** usando la clase **heartbeat** (), para ello crearemos el recurso **ClusterIP**.
Indicaremos un recurso (**ClusterIP**), que arrancara mediante el script de apache **ocf** y el proveedor **heartbeat** con el recurso **IPaddr2** que es el de la configuración de red.

```
#pcs resource create ClusterIP ocf:heartbeat:IPaddr2 ip="192.168.50.254"
cidr_netmask=24
```

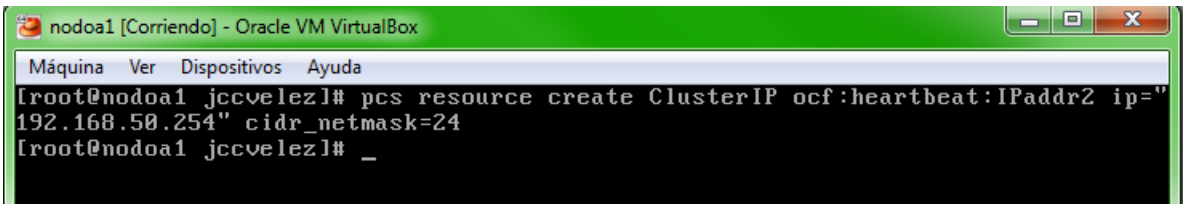


***Imagen 6.10. IP flotante del clúster.***

Ahora configuraremos el intervalo de comprobación entre los nodos de 30 segundos, este intervalo indicara cada cuando tiempo se comprobara el estado de los nodos y en caso de fallo se pasaran los recursos a otro nodo disponible.

```
#pcs resource op add ClusterIP monitor interval=30s
```

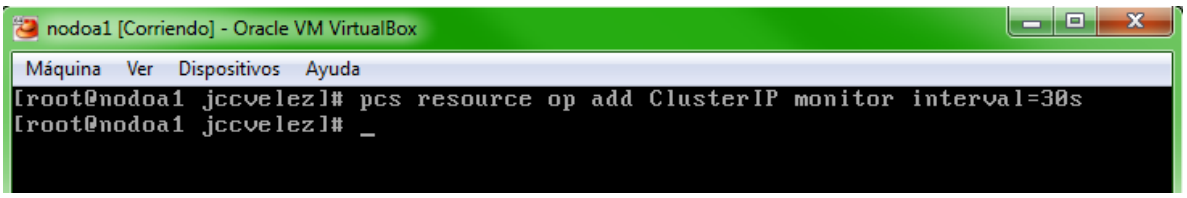


***Imagen 6.11. Intervalo de monitorización del clúster.***

Una vez realizado lo anterior ya tendremos configurado el clúster y podremos realizar ping a la dirección IP flotante.

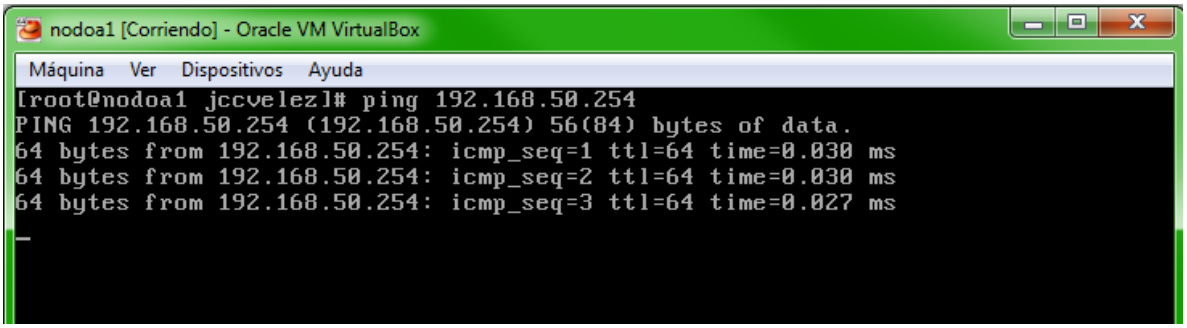


***Imagen 6.12. Comprobacion dela IP flotante.***

Dado que queremos ofrecer alta disponibilidad en un servicio web, instalaremos Apache para servir las páginas web, dicho servicio estará controlado por **pacemaker** que será el que envíe el recurso a otro nodo en caso de que **corosync** le indique que uno de los nodos activos (el que ofrece la web) falle.

Instalaremos apache (en ambos nodos).

```
#yum install httpd
```

Una vez instalado configuraremos un archivo de configuración creando un host virtual y escuchando de forma local para comprobar el estado de apache y en caso de fallo informar a **pacemaker** del fallo de forma local.

```
#nano /etc/httpd/conf.d/serverstatus.conf
```

```
Listen 127.0.0.1:80
 <Location /server-status>
 SetHandler server-status
 Order deny,allow
 Deny from all
 Allow from 127.0.0.1
 </Location>
```

Ahora configuraremos apache para que escuche “Listen” en el puerto 80 (por defecto) en ambos nodos.

```
#sed -i 's/Listen/#Listen/' /etc/httpd/conf/httpd.conf
```

Ahora reiniciaremos apache en ambos nodos.

```
#systemctl restart httpd
```

Ahora comprobaremos que funciona obteniendo el contenido del archivo.

```
#wget http://127.0.0.1/server-status
```


***Imagen 6.13. Archivo de comprobación de estado del clúster.***

Ahora crearemos un archivo index.html en /var/www/html con un contenido que lo diferencia del nodoa2 para identificar que nodo funciona en cada momento.

```
#nano /var/www/html/index.html
```

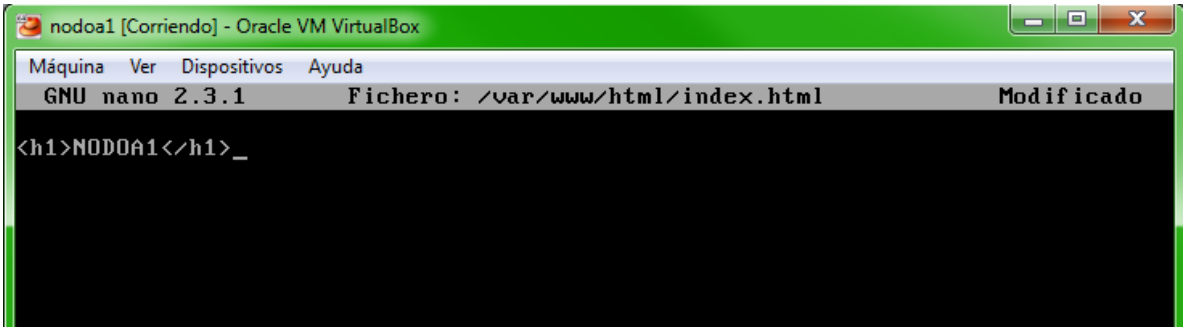


***Imagen 6.14. Contenido HTML del nodo.***

NOTA: Posteriormente todos los nodos de ambos cluster tendrán el mismo contenido gracias a la replicación **GlusterFS.**

Ahora añadiremos la dirección IP compartida como IP de escucha para apache y el puerto 80.

```
#echo "Listen 192.168.50.254:80"|sudo tee --append /etc/httpd/conf/httpd.conf
```

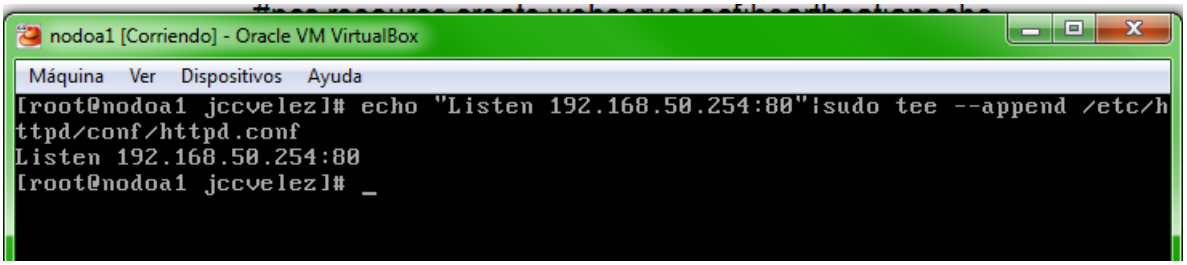


*Imagen 6.15. Puerto e IP de escucha de apache.*

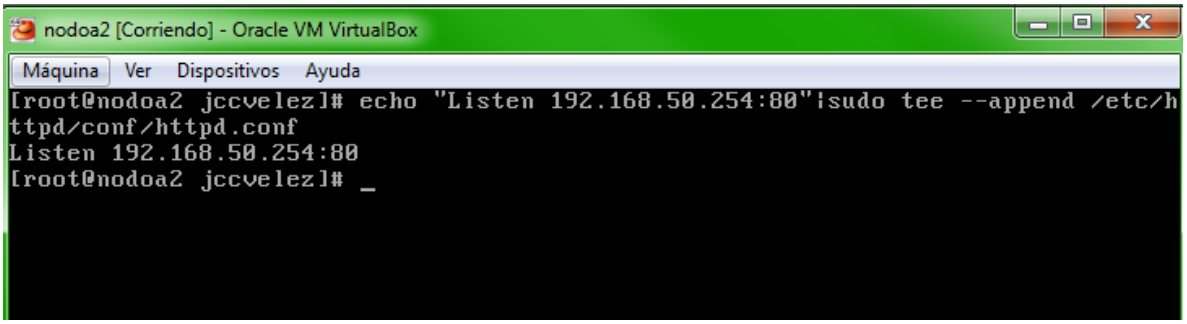


*Imagen 6.16. Puerto e IP de escucha de apache.*

Ahora añadiremos el recurso apache al clúster (únicamente lo haremos en el nodoa1) usando el script ocf de apache, la clase heartbeat y el proveedor apache indicando el archivo de configuración a través del cual se comprobara el estado de apache en el nodo con un intervalo de comprobación de 1 minuto.

```
#pcs resource create webserver ocf:heartbeat:apache configfile=/etc/httpd/conf/httpd.conf statusurl="http://localhost/server-status" op monitor interval=1min
```

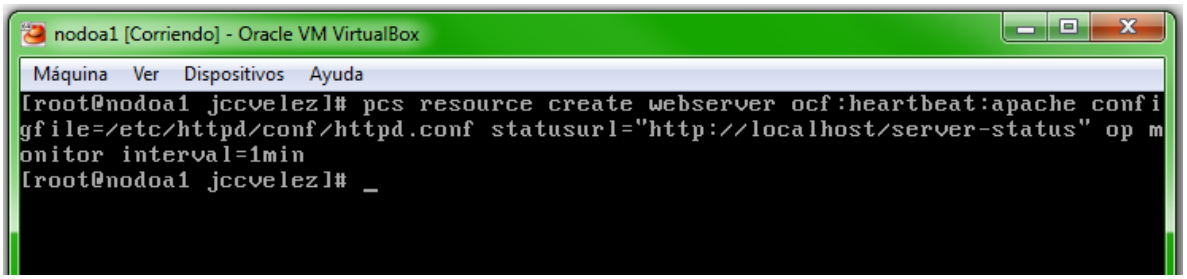


*Imagen 6.17. Recurso apache.*

Ahora añadiremos el recurso **webserver** creado al recurso ClusterIP para que forme parte del clúster con valor **INFINITY**, que permite que el recurso no cambie de nodo, es decir, únicamente cambia cuando hay un error en el nodo.
Si queremos que cambie a el nodo con menos carga usaremos el valor **0**.

```
#pcs constraint colocation add webserver ClusterIP INFINITY
```

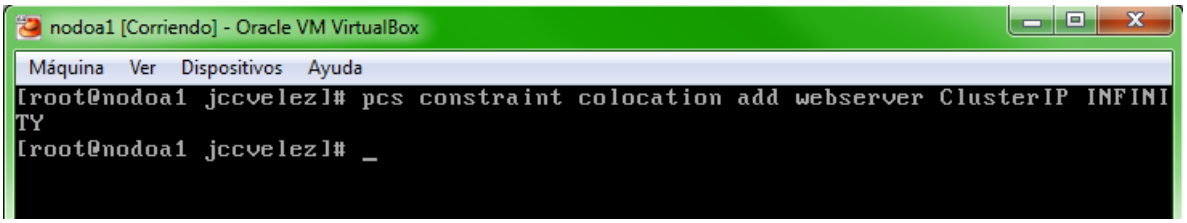


***Imagen 6.18. Añadir el recurso creado al cluster.***

También añadiremos otra propiedad para evitar que el servidor web inicie antes que el clúster, ya que deberemos especificar que el recurso **webserver** (apache) inicie una vez se ha iniciado el recurso **ClusterIP** (dirección IP flotante).

```
#pcs constraint order ClusterIP then webserver
```

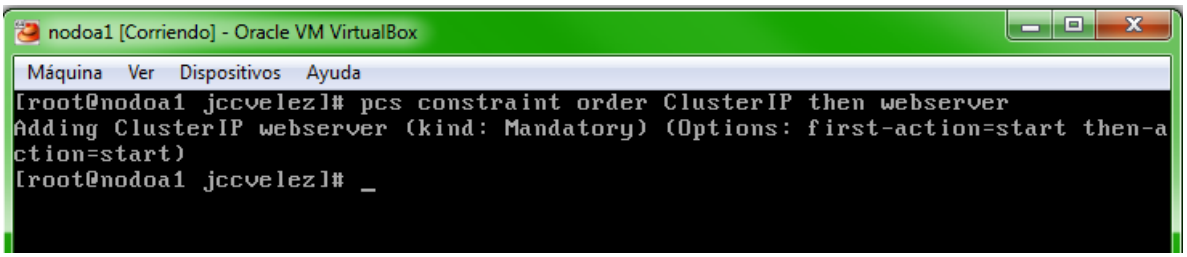


***Imagen 6.19. Propiedad del cluster.***

Ahora estableceremos que nodo será el preferido, ya que si disponemos de máquinas con diferente características deberemos indicar que la preferida es aquella con más recursos indicándole el porcentaje.

```
#pcs constraint location webserver prefers nodoa1=50
```

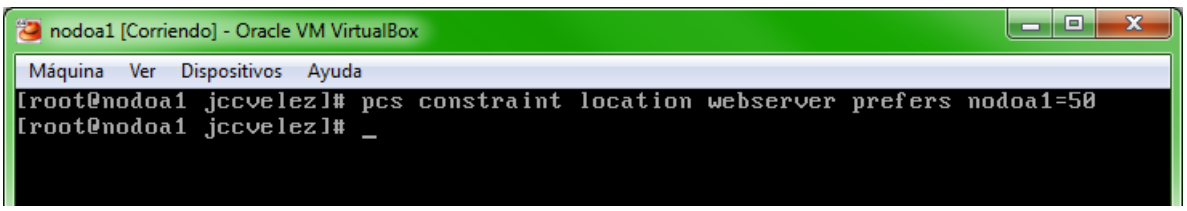


***Imagen 6.20. Nodo preferido.***

Finalmente pararemos el clúster y posteriormente iniciaremos todos los nodos del clúster.

```
#pcs cluster stop --all && sudo pcs cluster start –all
```

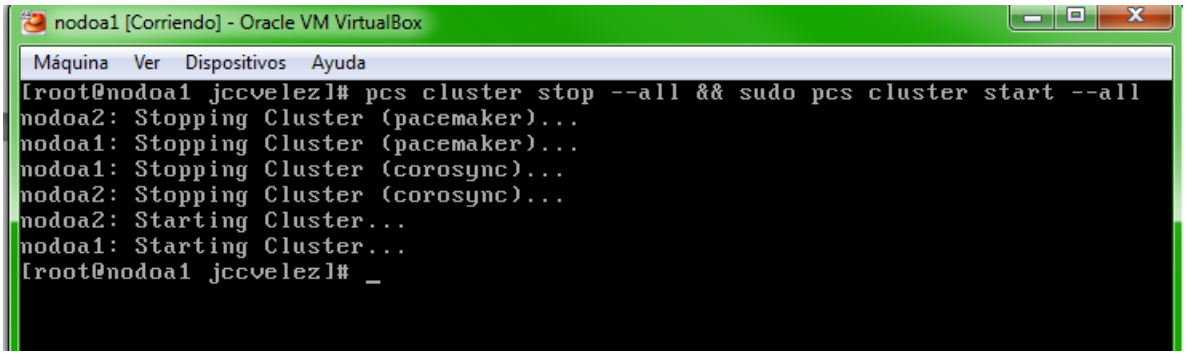


***Imagen 6.21. Inicio de los nodos del clúster.***

Ahora habilitaremos al inicio los servicios.

```
#systemctl enable pcsd
#sudo systemctl enable corosync
#sudo systemctl enable pacemaker
```

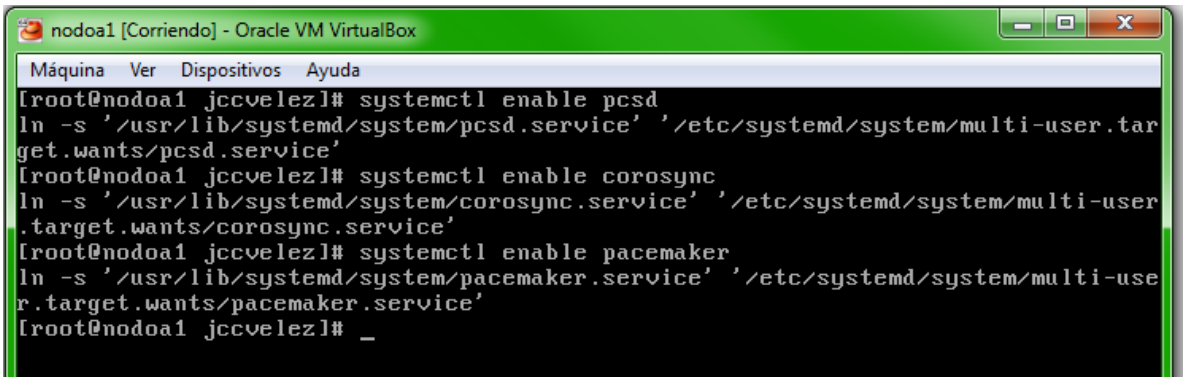


***Imagen 6.22. Habilitar servicios.***

Para que arranquen los servicios al iniciar los servidores (y evitar problemas) crearemos un script en **/etc/init.d/clusterinit.sh** en ambos nodos.

```
systemctl start corosync
systemctl start pcsd
pcs cluster start
```

Finalmente le daremos permisos de ejecución.

```
#chmod +x /etc/init.d/clusterinit.sh
```

Llegados a este punto ya tendremos configurado el clúster de servidores web.

# 7. Implementación y configuración de un sistema de balanceo de carga

Para dotar de seguridad y alta disponibilidad a los accesos a nuestros servicios implantaremos un sistema de Balanceo de Carga, cuya función es de enviar las peticiones provenientes del exterior hacia cada uno de los clusters anteriormente configurados (A y B), para obtener redundancia en el balanceo de carga dispondremos de dos balanceadores de carga con una dirección IP flotante usando **KeepAlived** y en caso de que uno de los balanceadores de carga falle entrara en funcionamiento el otro balanceador con la misma dirección IP ya que se comparten entre ambos.

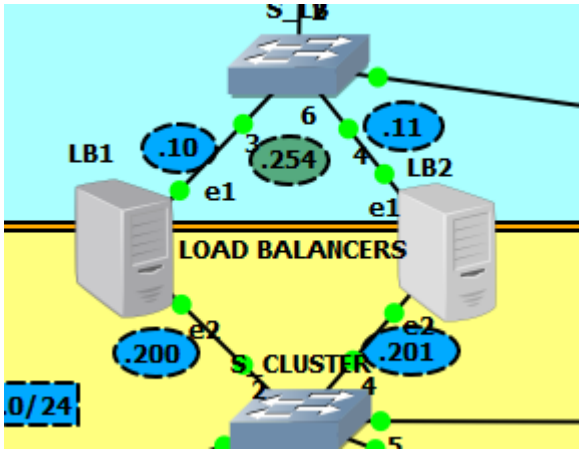


***Imagen 7.1. Esquema de balanceadores de carga.***

Existen diferentes tipos de balanceadores de carga en función de la capa del modelo OSI que se use.

**Capa 7**

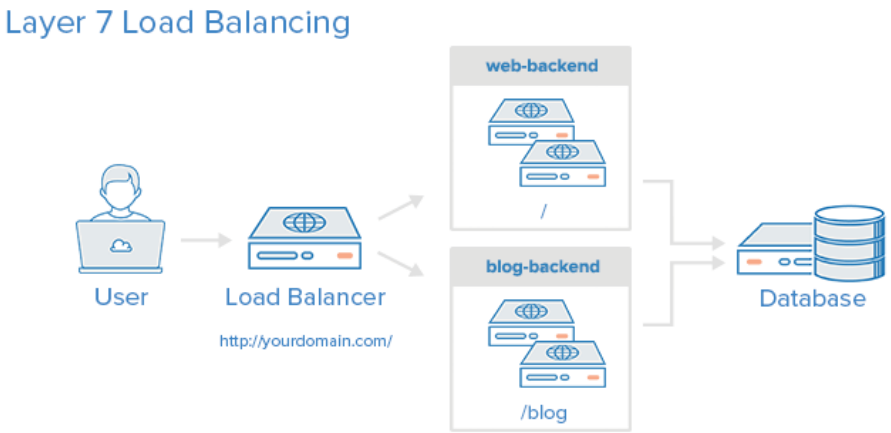


***Imagen 7.2. Balanceo de Capa 7.***

La función del equilibrio de la capa 7 (aplicación) es la de reenviar las peticiones de los usuarios en función del contenido de la solicitud del usuario observando la cabecera, lo que permite redirigir a varios servidores usando el mismo dominio, un ejemplo es el dominio **google.com/maps** que reenviara las peticiones a los servidores de mapas y el dominio **google.com/mail** que enviara la petición al servidor de correo.

## Capa 4

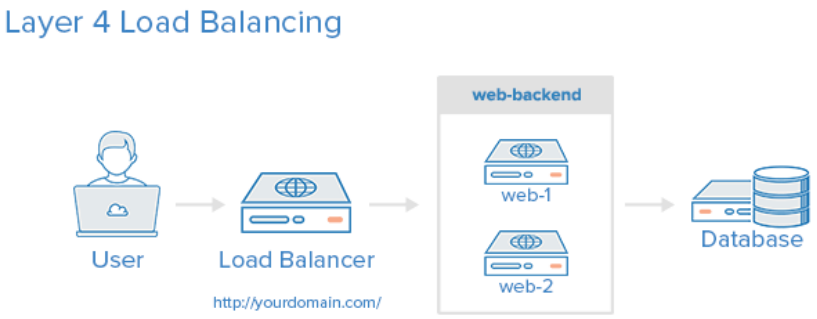


***Imagen 7.3. Balanceo de Capa 4.***

La función del equilibrio de la capa 4 (transporte) es la de reenviar las peticiones en función del dominio, subdominio o puerto que los usuarios soliciten, **este tipo de balanceo será el que usaré**.

Instalaremos la última versión disponible a la hora de realizar este proyecto, añadiremos los repositorios PPA, realizare la instalación en LB2, ya que en LB1 se realizaran los mismos pasos.

```
$sudo add-apt-repository ppa:vbernat/haproxy-1.5
```

Actualizamos el sistema para incluir el nuevo repositorio agregado.

```
$sudo apt-get update
```

Una vez actualizado el sistema instalaremos haproxy, para ello ejecutamos.

```
$sudo apt-get install haproxy
```

NOTA: La instalación la realizaremos en ambos Balanceadores (LB1 Y LB2)

Ahora configuraremos el balanceador LB1, para ello editamos el archivo de configuración **/etc/hacproxy/haproxy.cfg** y añadiremos la sección que será el frontend y el BackEnd (en realidad tendremos 2 BackEnd uno para los clusters y otro para el monitor del clúster).

En los “BackEnd” especificaremos los servidores a los cuales se reenviaran las peticiones que provengan del exterior (es decir, que provengan del FrontEnd), por lo que ofreceremos una seguridad de acceso, ya que el usuario no accede directamente al servidor web, si no que accede al Balanceador de carga (que actúa como proxy inverso) y el balanceador es el que accede al recurso y lo envíe al usuario que realizo la petición.

Añadimos lo siguiente al archivo de configuración.

```
option forwardfor
option http-server-close
```

Con estas opciones indicamos que use la opción de reenviar las peticiones que le llegan.

Ahora crearemos el FrontEnd que escuchara en la dirección IP externa de **KeepAlived** que configuraremos posteriormente, es decir en la dirección IP **192.168.20.254** que tendrán disponible ambos balanceadores.

Para ello incluimos la sección **frontend** y le damos un nombre, ya que podemos tener varias secciones, en mi caso se llamara **frontend clusterfrontend** y dentro de esta especificaremos cual será la dirección IP a través de la cual escuchara las peticiones para reenviarlas.

```
bind 192.168.20.254:80
```

Habilitaremos el **stats** y le asociaremos un usuario y una password para el inicio de sesión a través del navegador, que es un panel web de monitorización del balanceador donde podemos observar el estado de los FrontEnd y BackEnds, peticiones, ancho de banda usado, etc.

```
stats enable
stats auth jccvelez:inves
```

Ahora crearemos los dominios, para ello crearemos ACL’s ya que en nuestro caso tenemos 3 dominios, aunque 2 de ellos se comportan de igual manera tendremos el dominio **josecristian.net** y el subdominio **www.josecristian.net** que apuntaran al mismo backend, por otro lado para monitorizar el clúster y los balanceadores de carga habilitaremos otro dominio llamado **clustercp.josecristian.net** cuyo backend será diferente ya que al entrar al este ultimo la petición se redirigirá al servidor **MONITOR** que es el que posteriormente dispondrá de la aplicación PHP que permitirá la monitorización del clúster y del Balanceador de carga.

Crearemos una ACL llamada **cluster** para el dominio **josecristian.net** y otra llamada **clusterwww** para el dominio **www.josecristian.net**, posteriormente crearemos otra ACL llamada **clustercp** para el subdominio **clustercp.josecristian.net**, para ello en la sección **clusterfrontend** insertamos las ACL’s.

La función de las ACL permite que si un usuario entra a través de un dominio se le redirija a un “BackEnd” determinado.

```
acl cluster hdr(host) –i josecristian.net
acl clusterwww hdr(host) –i www.josecristian.net
acl clustercp hdr(host) –i clustercp.josecristian.net
```

Ahora deberemos indicar a que backend lo reenviaremos, a pesar de que los backends los vamos a crear después definiremos como se llamara el backend al que redireccionaremos cada ACL, para ello indicaremos que backend tiene que usar cuando se cumple una ACL de la siguiente manera.

```
use_backend backcluster if cluster
use_backend backcluster if clusterwww
use_backend backclustercp if clustercp
```

Estamos indicando usar un backend (ahora depues los definiremos) si se cumple una ACL.

Ahora crearemos los backend, como hemos visto antes tenemos dos backend, uno de ellos para el **cluster** y otro de ellos para el **clustercp**, para ello crearemos cada backend como diferentes secciones de la siguiente manera.

```
backend backcluster
option httpclose
option forwardfor
server clustera 192.168.50.254:80 check
server clusterb 192.168.50.253:80 check

backend backclustercp
option httpclose
option forwardfor
server monitor 192.168.50.190:80 check
```

Como vemos hemos creado los dos BackEnd y hemos indicado que realice la redirección a un servidor y otro de los que componen cada backend, el orden de los servidores se respetara en las peticiones, por defecto HAProxy realiza un balaceo de carga **RoundRobin**, es decir, cada petición que llega la envía a un servidor diferente, otro tipo de balanceo llamado **leastconn** cuyo funcionamiento se basa en enviar las peticiones al clúster o servidor con menos peticiones y por lo tanto más bajo de carga, existe un último tipo de balanceo denominado **source** cuya funcionamiento consiste en redirigir una misma dirección IP al mismo servidor, por lo que el usuario navegará siempre en el mismo servidor.

Para modificar el tipo de balanceo deberemos modificar el parámetro **balance** en la sección **defaults** indicando el tipo de balanceo deseado.

El archivo **haproxy.cfg** del balanceador LB1 queda así:

```
global
        log /dev/log local0
        log /dev/log local1 notice
        chroot /var/lib/haproxy
        stats socket /run/haproxy/admin.sock mode 660 level admin
        stats timeout 30s
        user haproxy
        group haproxy
        daemon

        # Default SSL material locations
        ca-base /etc/ssl/certs
        crt-base /etc/ssl/private

        # Default ciphers to use on SSL-enabled listening sockets.
        # For more information, see ciphers(1SSL). This list is from:
        #  https://hynek.me/articles/hardening-your-web-servers-ssl-
ciphers/
        ssl-default-bind-ciphers
ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES12
8:DH+AES:ECDH+3DES:DH+3DES:RSA+AESGCM:RSA+AES:RSA+3DES:!
aNULL:!MD5:!DSS
        ssl-default-bind-options no-sslv3

defaults
        log     global
        mode http
        balance roundrobin
        option forwardfor
        option http-server-close
        option          httplog
        option          dontlognull
      timeout connect 5000
      timeout client  50000
      timeout server  50000
        errorfile 400 /etc/haproxy/errors/400.http
        errorfile 403 /etc/haproxy/errors/403.http
        errorfile 408 /etc/haproxy/errors/408.http
        errorfile 500 /etc/haproxy/errors/500.http
        errorfile 502 /etc/haproxy/errors/502.http
        errorfile 503 /etc/haproxy/errors/503.http
        errorfile 504 /etc/haproxy/errors/504.http
frontend clusterfrontend
        bind 192.168.20.254:80
        stats enable
        stats auth jccvelez:inves
        #dominios
        acl cluster hdr(host) -i josecristian.net
```

```
        acl clusterwww hdr(host) -i www.josecristian.net
        acl clustercp hdr(host) -i clustercp.josecristian.net
        #desvios
        use_backend backcluster if cluster
        use_backend backcluster if clusterwww
        use_backend backclustercp if clustercp
backend backcluster
        option httpclose
        option forwardfor
        server clustera 192.168.50.254:80 check
        server clusterb 192.168.50.253:80 check
backend backclustercp
        option httpclose
        option forwardfor
        server monitor 192.168.50.190:80 check
```

Realizaremos el mismo proceso en el balanceador LB2, el archivo de configuración será el mismo ya que como la dirección IP esta compartida entre ambos balanceadores se usara el mismo archivo de configuración, por lo que lo podremos copiar del **LB1** al **LB2**

**KeepAlived**

Una vez instalado y configurado HAProxy implementaremos KeepAlived en ambos Balanceadores, la función de este paquete es la de proveer una dirección IP flotante compartida para ambos Balanceadores, de tal manera que al fallar uno de ellos automáticamente la dirección IP cambia al otro y otros balanceadores y así seguir dando el servicio.

Para ello lo instalamos ambos nodos.

```
$sudo apt-get install keepalived
```

Realizaremos el mismo proceso en el balanceador LB2 instalando KeepAlived.

Una vez instalado configuraremos la IP virtual que será la dirección IP **192.168.20.254** y la prioridad de KeepAlived en cada Balanceador, LB1 tendrá prioridad **101** y **LB2** tendrá prioridad **100**, a mayor número mayor prioridad y si es MASTER (LB1) o BACKUP (LB2).

La configuración será la misma en ambos balanceador a excepción de la prioridad y si es MASTER o BACKUP, para ello crearemos el archivo **/etc/keepalived/keepalived.conf** con la configuración.

```
vrrp_script chk_haproxy {
  script "killall -0 haproxy" # comprobamos el proceso de keepalived
  interval 2 # tiempo de comprobacion, 2 segundos
  weight 2 # si esta correcto le damos 2 puntos
}
vrrp_instance VI_1 {
  interface eth1 # interfaz en la que actua keepalived
  state MASTER # LB1 MASTER,LB2 es BACKUP
  virtual_router_id 51
  priority 101 # prioridad, contra más alto más prioridad
  virtual_ipaddress {
    192.168.20.254 # ip virtual
  }
  track_script {
    chk_haproxy  # seguimiento
  }
}
```

La configuración del archivo **/etc/keepalived/keepalived.conf** del balanceador **LB2** será la siguiente:

```
vrrp_script chk_haproxy {
  script "killall -0 haproxy" # comprobamos el proceso de keepalived
  interval 2 # tiempo de comprobacion, 2 segundos
  weight 2 # si esta correcto le damos 2 puntos
}

vrrp_instance VI_1 {
  interface eth1 # interfaz en la que actua keepalived
  state BACKUP # LB1 MASTER,LB2 es BACKUP
  virtual_router_id 51
  priority 100 # prioridad, contra más alto más prioridad
  virtual_ipaddress {
    192.168.20.254 # ip virtual
  }
  track_script {
    chk_haproxy  # seguimiento
  }
}
```

# 8. DNS (Sistema de Nombres de Dominio)

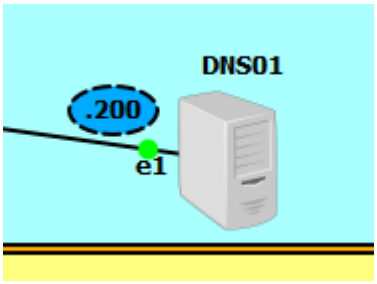


*Imagen 8.1. Servidor DNS.*

Ahora instalaremos un servidor DNS que será el encargado del dominio, lo instalaremos en un servidor independiente llamada **dns01**, será instalado sobre Ubuntu Server 14.04.2 usando BIND9.

Crearemos la zona **josecristian.net** con los registros tipo A para **www.josecristian.net y monitor.josecristian.net** apuntando a la dirección IP flotante del balanceador (192.168.20.254) que será el encargado de redirigirla al servidor correspondiente.

Configuraremos la zona directa editando el archivo **/etc/bind/named.conf**

```
zone “josecristian.net” {
type master;
file “/etc/bind/josecristian.net”;
};
```

Ahora editamos el archivo de opciones de BIND9 y permitimos las consultas desde cualquier origen añadiendo al archivo **/etc/bind/named.conf.options** lo siguiente:

```
allow-query{any;};
```

Una vez creada la zona directa crearemos el archivo **/etc/bind/josecristian.net** en el que configuraremos los registros SOA,NS, A y CNAME (www), el archivo de configuración quedará así:

```
$TTL 604800
@     IN     SOA   josecristian.net.   jose.josecristian.net. (
                          2         ;Serial
                     604800         ;Refresh
                      86400         ;Retry
                    2419200         ;Expire
                     604800         ;Negative Cache TTL
)
@     IN     NS    josecristian.net.
@     IN     A     192.168.20.254
```

**www IN CNAME josecristian.net.**
**clustercp IN A 192.168.20.254**

Una vez configurado el servidor DNS configuraremos el archivo **/etc/resolv.conf** para que el servidor se consulte así mismo, añadiendo:

```
nameserver 127.0.0.1
```

Ahora reiniciamos BIND9.

```
$sudo service bind9 restart
```

Si comprobamos la resolución veremos que nos resuelve correctamente.

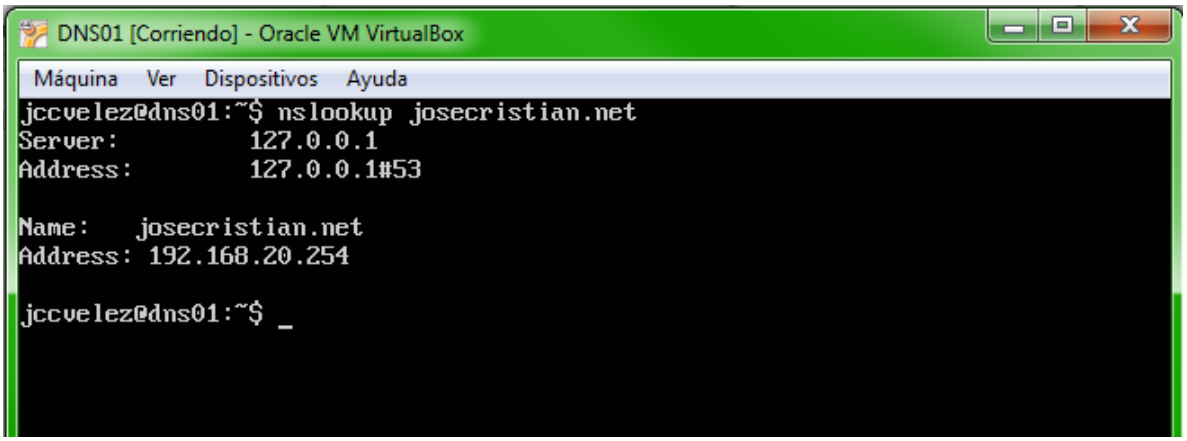


***Imagen 8.2. Comprobación de resolución DNS.***

# 9. Replicación de Datos


***Imagen 9.1. Logo GlusterFS.***

La replicación de datos consiste en disponer de los mismos archivos en todos los servidores que componen el clúster, es decir, **nodoa1 nodoa2 nodob1 y nodob2**, para ello usaremos **GlusterFS**, lo usaremos en forma de RAID1 pero a nivel de red, es decir, replicar el contenido de un disco duro o partición entre todos los discos duros o particiones deseados, este servicio es más versátil que DRBD y escalable ya que DRBD permite un máximo de 4 nodos.

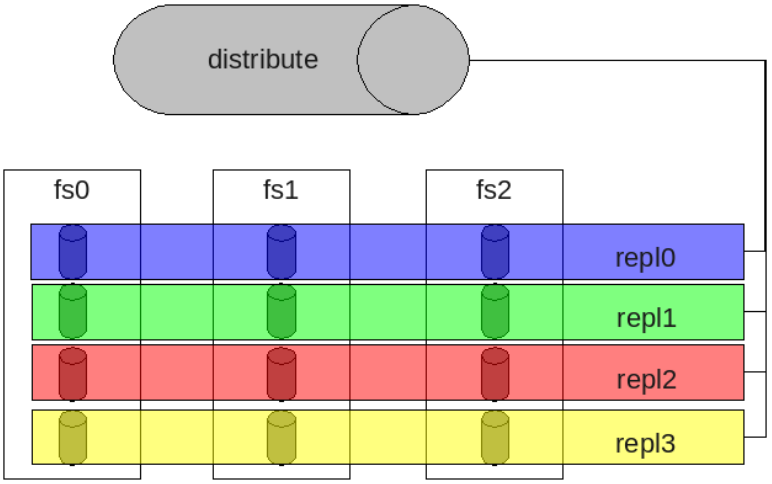


***Imagen 9.2. Esque de replicación GlusterFS.***

Añadiremos un segundo disco duro para DRBD en cada uno de los nodos de 5GB (ya que estamos en un entorno de virtualización), este será reconocido como **/dev/sdb1**.

NOTA: Realizaremos lo mismo en todos los nodos que forman el clúster A y B.

Una vez añadidos los discos realizaremos una partición de todo el almacenamiento en **/dev/sdb1** en cada uno de los nodos usando el comando **fdisk**.

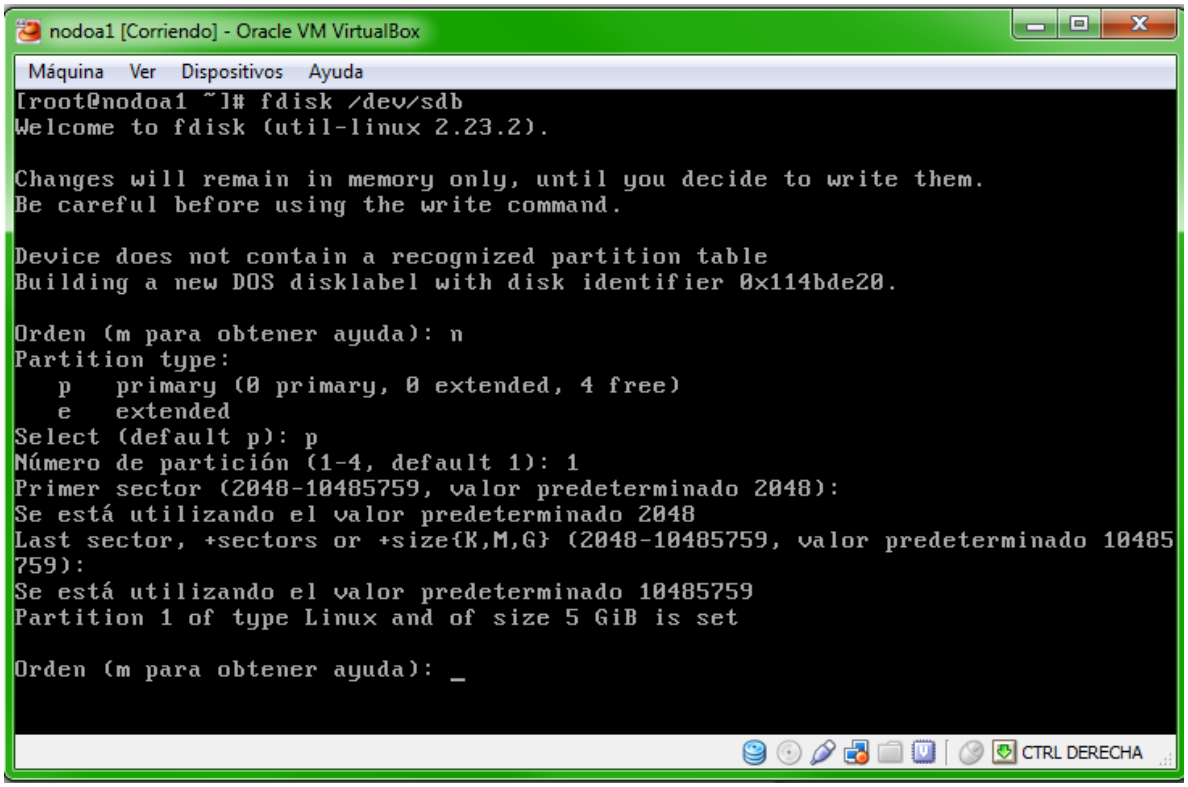


***Imagen 9.3. Creación de la partición.***

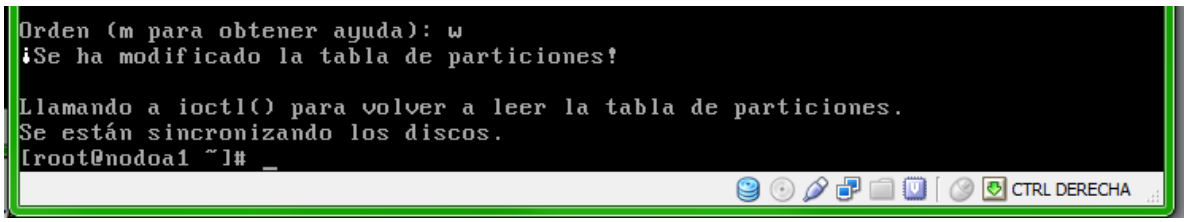


***Imagen 9.4. Escribir los cambios en la partición.***

Le daremos un sistema de ficheros, será XFS ya que tiene más estabilidad, menos errores y más almacenamiento que ext4.

```
#mkfs.xfs /dev/sdb1
```

Ahora crearemos un carpeta donde montaremos la partición, la carpeta se llamara **/gfs** y montaremos la carpeta (en todos los nodos).

```
#mkdir /gfs
#mount /dev/sdb1 /gfs
```

Para que arranque con el sistema añadiremos una entrada en el archivo **/etc/fstab**.

```
/dev/sdb1    /gfs    xfs     defaults 0 2
```

Como indicamos en la configuración de red disponemos de una red específicamente para la replicación de datos y así disponer del ancho de banda disponible para los datos y así evitar que si otra red falla no afecte a la replicación de los datos.

Configuraremos el nombre de los nodos para la replicación de datos en cada uno de los nodos con la dirección IP correspondiente, editaremos el archivo **/etc/hosts** (de cada uno de los nodos) y añadiremos las direcciones IP de cada uno de los nodos para GlusterFS identificando cada nodo como **nodogfsXY** (donde X es la letra de nodo e Y es el número de nodo) lo siguiente:

```
172.17.50.10        nodogfsa1
172.17.50.11        nodogfsa2
172.17.50.15        nodogfsb1
172.17.50.16        nodogfsb2
```

Ahora comprobaremos la resolución haciendo ping.

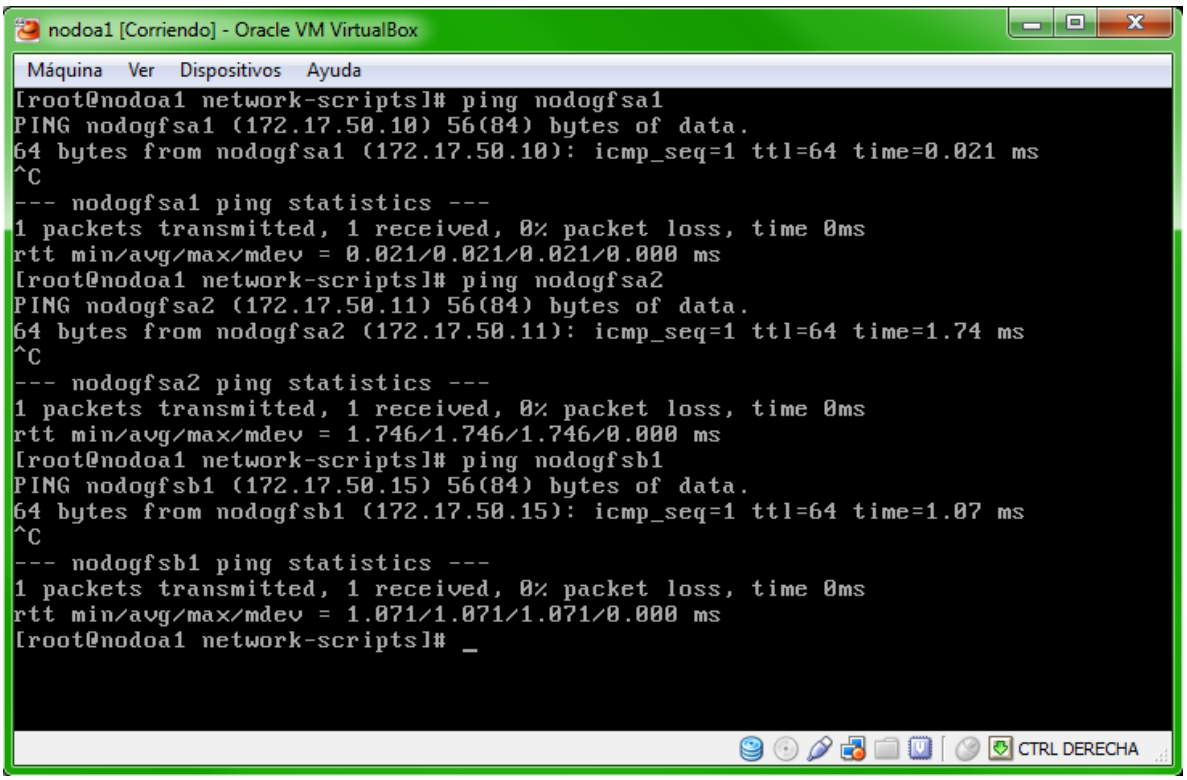


***Imagen 9.5. Comprobacion del ping.***

Una vez configurado el sistema instalaremos **GlusterFS** en todos los nodos, primero instalaremos el repositorio EPEL de CentOS e instalaremos el paquete **userspace-rcu-devel** que es necesario para instalar **GlusterFS**.

Para ello ejecutaremos:

```
$sudo yum install epel-release
$sudo yum install userspace-rcu-devel
```

Posteriormente descargaremos el repositorio de GlusterFS e instalaremos el paquete **glusterfs, glusterfs-fuse** (cliente glusterfs), y **glusterfs-server**

```
$sudo wget -P /etc/yum.repos.d
http://download.gluster.org/pub/gluster/glusterfs/LATEST/EPEL.repo/glusterfs-epel.repo
$sudo yum -y install glusterfs glusterfs-fuse glusterfs-server
```

Una vez instalado iniciaremos el servicio:

```
$sudo service glusterd start
```

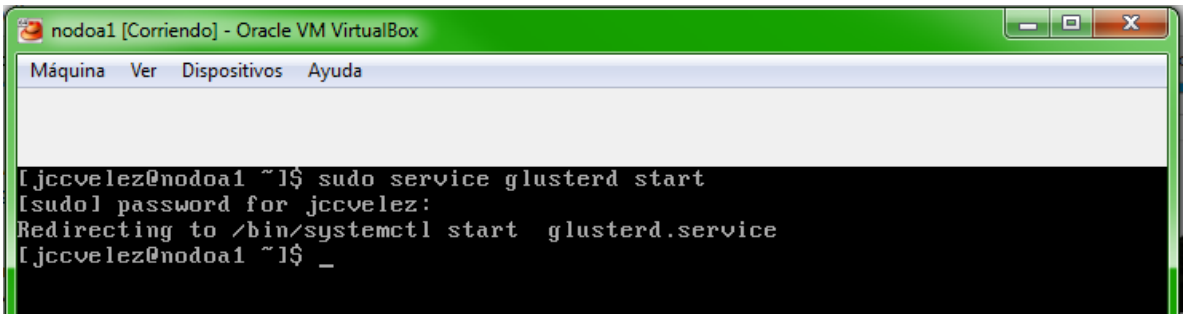


***Imagen 9.6. Inicio del servicio.***

Para que el servicio inicie con el sistema ejecutaremos:

```
#chkconfig glusterd on
```

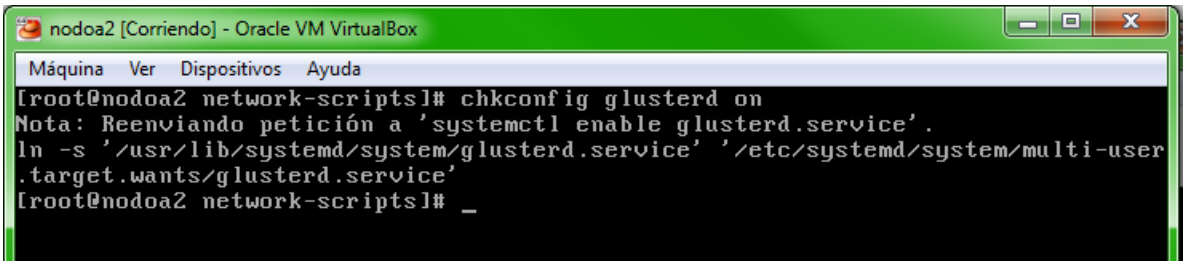


***Imagen 9.7. Habilitación del servicio.***

Ahora configuraremos los nodos para el **GlusterFS**. Realizaremos un "escaneo" para añadir los nodos que formaran parte del **GlusterFS.**

Usaremos el siguiente comando que nos dará un mensaje **success** si esta todo correcto.

```
#gluster peer probe nombre_nodo
```

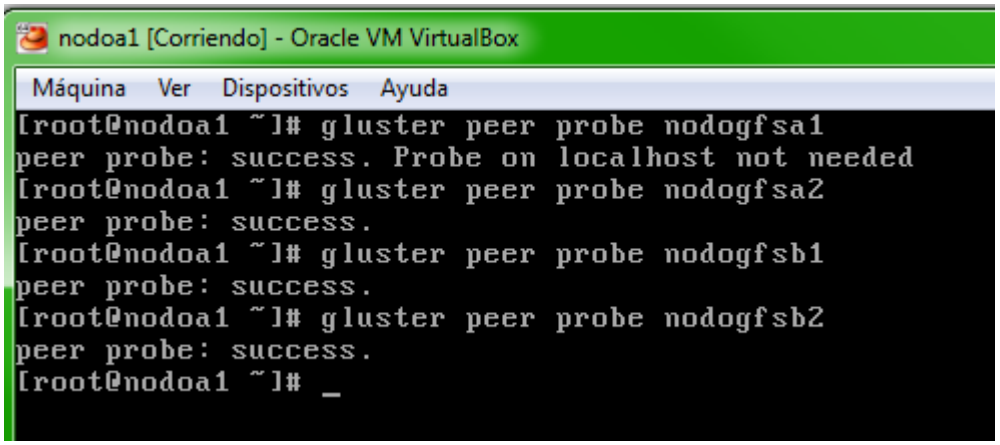


***Imagen 9.8. Comprobación de conexión con los servidores de GlusterFS.***

Ahora comprobaremos que los nodos se han añadido correctamente

```
#gluster peer status
```

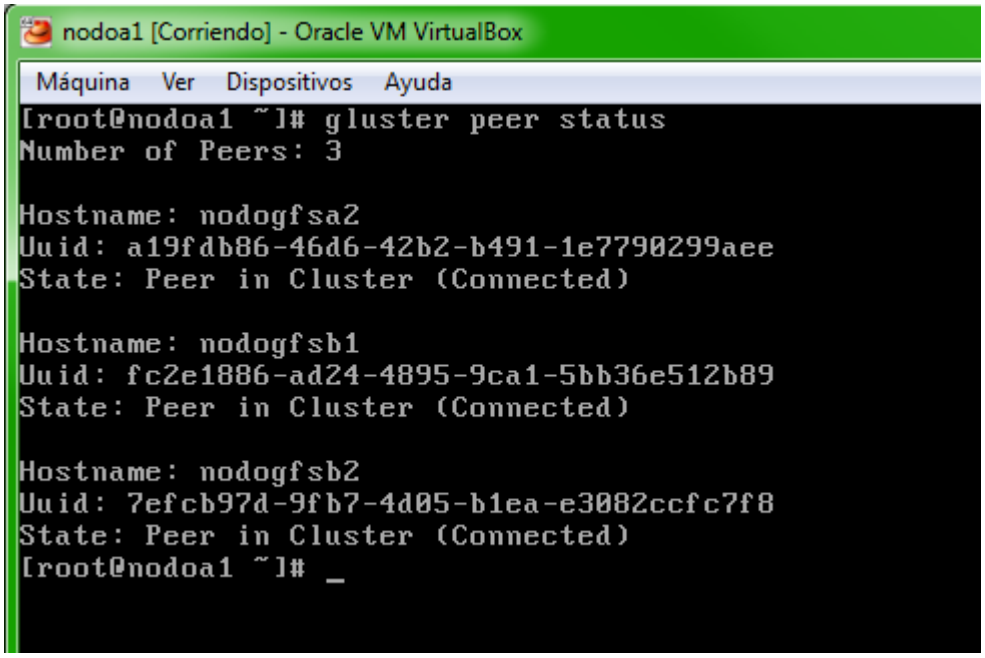


***Imagen 9.9. Estado de los servidores, conexión.***

NOTA: Vemos que hay 3 nodos conectados, a parte del nodo en el que ejecutamos el comando.

Ahora crearemos el volumen llamado **gfsweb** en modo **replica** e indicaremos las rutas de los nodos donde se replicaran los archivos, antes deberemos crear una carpeta bajo la carpeta raíz **/gfs** ya que no se permite crear el volumen en la raíz, la carpeta se llamara **/gfs/www** (la crearemos en cada uno de los nodos).

Para crear el volumen ejecutaremos (únicamente en el nodo1):

```
#gluster volume create gfsweb replica 4 transport tcp
nodogfsa1:/gfs/www nodogfsa2:/gfs/www nodogfsb1:/gfs/www
nodogfsb2:/gfs/www
```

Parámetros:

- gfsweb: es el nombre del volumen
- replica: es el modo de GlusterFS, modo replicación
- 4: numero de nodos
- transport tcp: modo de envio de los archivos a través de tcp, por lo que se comprueban los archivos (ACK).
- nodogfsa1:/gfs/www: destino donde se montara el volumen en cada nodo

Una vez creado lo iniciaremos.

```
#gluster volume start gfsweb
```

NOTA: Si todo va bien nos dará el mensaje **success**.

Podemos comprobar los **bricks (**ladrillos, que es asi como se llama a cada uno de los nodos que forma parte de la replicación**)**.

```
#gluster volumen info
```

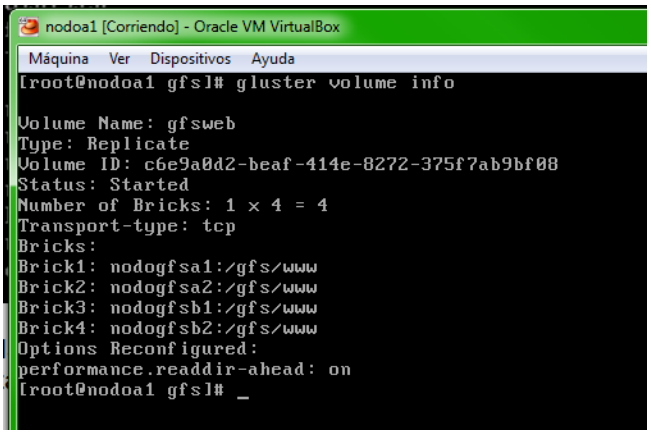


***Imagen 9.10. Informacion del volumen creado.***

Por seguridad no se puede acceder a este volumen directamente como si fuera una carpeta (es decir, ya que está montada en **/gfs/www**), para replicar archivos y acceder a ellos deberemos montar el volumen en una nueva carpeta, como será para apache montaremos el volumen **gfsweb** en **/var/www/html** (eliminaremos la carpeta **html** existente y crearemos otra llamada **html**) para poder tener el mismo contenido que ofrece apache en cada uno de los nodos del cluster.

Montaremos el volumen **gfsweb** (el tipo de sistema de archivo es **glusterfs**) en **/etc/fstab** para que se monte en el arranque de cada equipo en si mismo (cada nodo será su mismo punto de montaje para poder acceder al contenido).

```
localhost:/gfsweb /var/www/html glusterfs defaults,_netdev 0 0
```

Podemos comprobar que se ha montado correctamente.

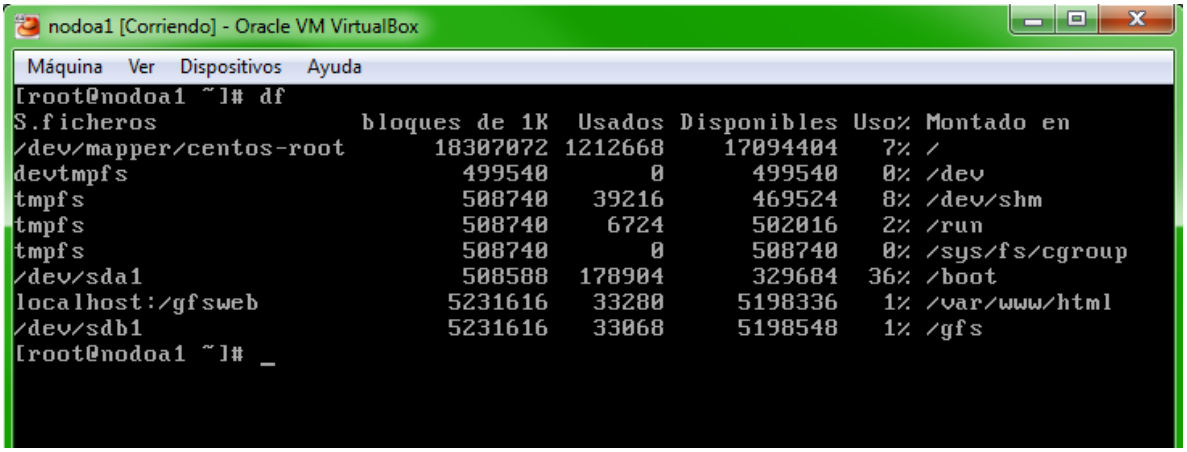


***Imagen 9.11. Comprobación del montado del volumen creado.***

NOTA: Realizaremos lo mismo en cada equipo.

Una vez montado el sistema de replicación con **GlusterFS** añadiremos el contenido que ofrecerá el servicio web de apache, para ello únicamente es necesario añadir los archivos en un nodo (da igual el que sea) y los archivos estarán replicados en todos los nodos.

Tendremos una página con información y una variable PHP que permitirá identificar el nodo que está ofreciendo la página web a través del nombre del equipo, usaremos la función **$gethostname()**, en “producción” no deberíamos identificar el nodo que está ofreciendo la web, pero en un entorno de “desarrollo” si es importante para verificar el funcionamiento.

El contenido del archivo será el siguiente:

```
<html>
<head>
<title>
WEB CLUSTER
</title>
</head>
<body bgcolor="#3686BE">
<center>
<div>
<h1>WEB ALOJADA EN EL CLUSTER REPLICADA USANDO
glusterFS</h1>
<?php echo "<h2>NODO: ".gethostname()."</h2>" ?>
</div>
</center>
</body>
</html>
```

Ahora si accedemos desde el equipo cliente ElementaryOS al dominio **josecristian.net** veremos la web y el nodo que nos está ofreciendo la web gracias a la variable **$gethostname()**.

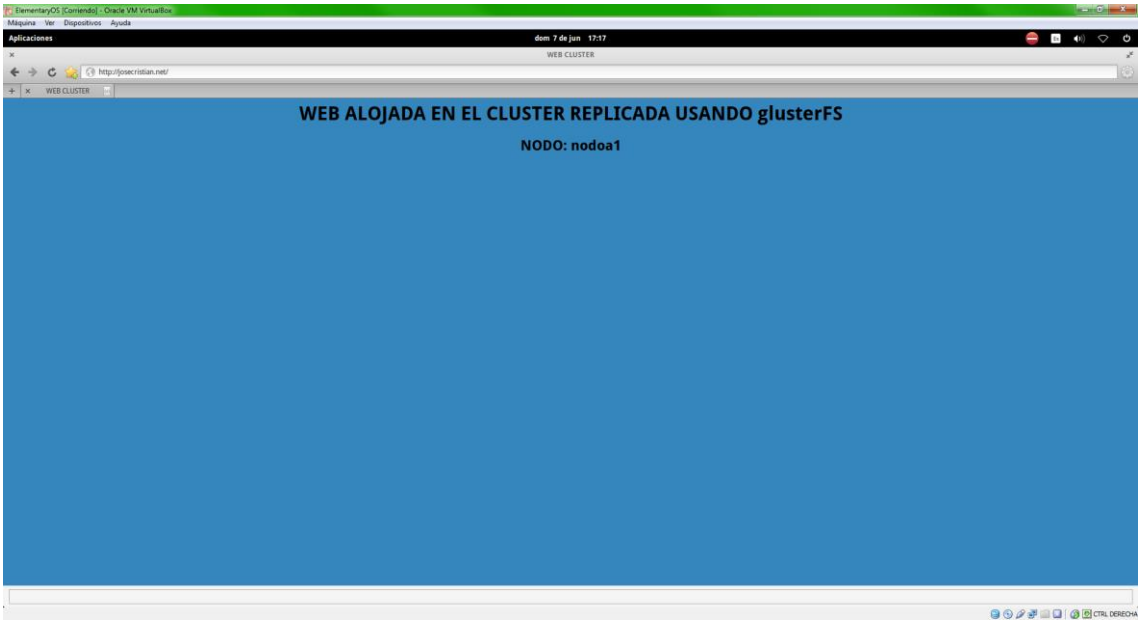


***Imagen 9.12. Comprobacion de la web.***

Si recargamos la web veremos que el nodo cambia ya que el balanceador reenvía la petición al nodo activo del otro Cluster.

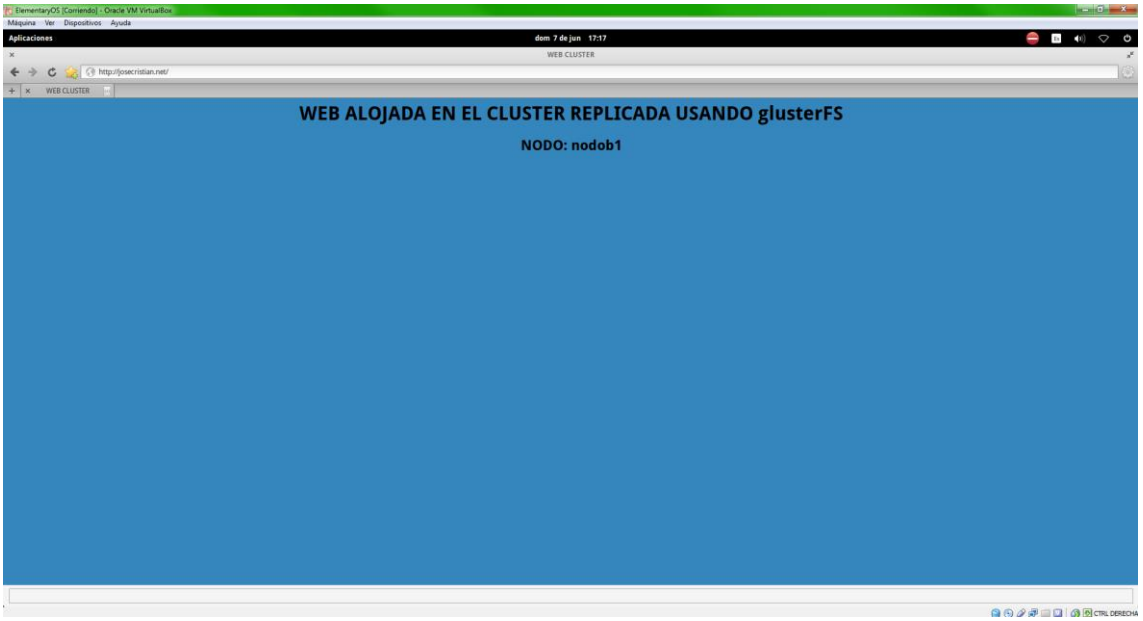


***Imagen 9.13. Comprobacion de la web.***

# 10. Sistema de Monitorización

La finalidad del sistema de monitorización es la de “controlar” los clusters y los nodos en caso de fallo de alguno de ellos, para ello tendremos una imagen de cada de cada nodo y en caso de estar todo correctamente mostrara una imagen de que todo está correctamente y en caso de fallo se mostrará una X en color ROJO, además cuando uno de los nodos falle se nos notificara a través de correo electrónico y se guardara en un log local (Punto 9.5).

El sistema de monitorización estará creado en lenguaje PHP, HTML5 y CSS3 y JavaScript.

También dispondremos de un sistema de estadísticas en la cual se nos mostrará a través de diferentes gráficos las caídas de cada nodo, caídas por fechas y la memoria RAM de cada clúster (nodo activo en el momento de cada clúster).

Dispondremos en el menú un enlace al sistema de estadísticas de HAproxy, dicho sistema nos indica los Balanceadores activos, número de solicitudes por destino, etc.

La Aplicación web PHP se ha realizado usando el patrón de diseño MVC (Modelo-Vista-Controlador), dicho patrón permite separar los datos que el usuario visualiza y por otro lado la aplicación que gestionar esos datos.

Para ello MVC separa la aplicación en 3 componentes:

- Modelo: Es el encargado de realizar los accesos a la información como base de datos, y de generar los datos que el usuario visualizara, usando funciones, clases, etc.
- Vista: Presenta los datos generados por el modelo al usuario, es decir, recoge los datos del modelo y les da un formato adeacuado para que el usuario pueda visualizar e interactuar con él.
- Controlador: Es el encargado de responder a los eventos del usuario, como presionar un botón, etc. para ello el controlador "conecta" el modelo con la vista.

A continuación se muestra la estructura de directorios y archivos de la aplicación **clusterCP**.

**--clustercp (Nivel 1) → Proyecto.**
**---app (Nivel 2)**
**-----controladores (Nivel 3)**
**--------crearusuario_controlador.php → Controlador de crear usuario.**
**--------estadisticas_cluster_controlador.php → Controlador de estadísticas clúster.**
**--------estado_cluster_controlador.php → Controlador del estado del clúster.**
**--------nuevo_cluster_controlador.php → Controlador de crear nuevo clúster.**
**-------estado_balanceador_controlador → Controlador de estado del LB.**
**-----db (Nivel 3)**
**--------conexion.php → Conexión a la BBDD.**
**-----includes (Nivel 3)**
**--------seguridad.php → Comprobación del logueo del usuario.**
**--------ssh.php →Archivo de direcciones IP y datos para la conexión MongoDB**
**-----modelos (Nivel 3)**
**--------crearusuario_modelo.php → Modelo de crear usuario.**
**--------estadisticas_cluster_modelo.php → Modelo de estadísticas clúster.**
**--------estado_cluster_modelo.php → Modelo del estado del clúster.**
**--------nuevo_cluster_modelo.php → Modelo de crear nuevo clúster.**
**--------estado_balanceador_modelo → Modelo de estado del balanceador.**
**-----vistas (Nivel 3)**
**--------crearusuario_vista.php → Vista de crear usuario.**
**--------estadisticas_cluster_vista.php → Vista de estadísticas clúster.**
**--------estado_cluster_vista.php → Vista del estado del clúster.**
**--------nuevo_cluster_vista.php → Vista de crear nuevo clúster.**
**--------estado_balanceador_vista → Vista de estado del balanceador.**
**---web (Nivel 2)**
**-----css(Nivel 3) → Estilos css del login.**
**--------estilo.css → Estilo css del sistema de login.**
**-----img (Nivel 3) → Imágenes.**
**--------clusterCP.JPG → Logo de clusterCP.**
**-----acceso.php → Comprobación de acceso a la BBDD (usuario y pass).**
**-----index.php → Archivo de login.**
**-----salir.php → Archivo para salir y destruir la sesión.**
**-----dashboard (Nivel 3) → Carpeta de aplicaciones como estadísticas, etc.**
**--------estado_cluster (Nivel 4) → Aplicación para monitorizar los clusters.**
**----------css (Nivel 5) → Estilos CSS.**

----------estilo.css → **Estilo CSS de la página de monitoriza el clúster.**
----------**img (Nivel 5) → Imágenes.**
----------error.png → **Imagen de nodo con error.**
----------ok.png →**Imagen de nodo correcto.**
----------server.png →**Imagen de servidor.**
----------index.php →**Página de inicio que muestra el estado de cada nodo.**
--------**estadisticas_cluster (Nivel 4) →Carpeta de la aplicación de estadísticas.**
----------index.php →**Página de inicio de estadísticas. Muestra las caídas.**
----------**css (Nivel 5) → Estilos CSS.**
----------estadisticas.css → **Estilo de estadísticas, cajas, etc.**
--------**balanceador(Nivel 4) →Carpeta de la aplicación del balanceador.**
----------index.php →**Página de inicio de la aplicación del balanceador.**
--------**nuevo_cluster (Nivel 4) →Carpeta de la aplicación de crear nuevo clúster.**
----------index.php →**Página de inicio de crear nuevo clúster.**
--------**nuevo_usuario (Nivel 4) →Carpeta de la aplicación de nuevo usuario.**
----------index.php → **Formulario para crear usuarios.**
--------**includes (Nivel 4) →Archivos de uso común en las aplicaciones.**
----------sidebar.php →**Archivo de menú izquierdo para navegar.**
--------**css (Nivel 4) →Estilos CSS.**
----------form.css → **Estilo de formulario de creación de usuarios.**
----------general.css →**Estilo general de fondo, posición de menú, etc.**
--------**js (Nivel 4) →Estilos CSS.**
----------validador_nuevo_cluster.js → **Validar form nuevo cluster.**
----------validador_nuevo_usuario.js → **Validar form nuevo usuario.**

**Leyenda de Colores: Directorios archivos imágenes**

NOTA: En apache el directorio padre por defecto será **/var/www/html/web** para que los usuarios no tengan acceso **directamente** a los modelos, vistas y controladores del servidor.

## 10.1 Base de datos MongoDB


***Imagen 10.1.1. Logo de MongoDB.***

MongoDB es una base de datos NoSQL (Non Only SQL – No Solo SQL) orientada a documentos, es decir, la información no se guarda en tablas o registros como en MySQL si no en documentos en formato BSON (representación binaria de JSON).

Este tipo de Base de Datos es rápida, fiable y fácil de escalar debido a que no sigue una estructura fija (no como en MySql que se definen columnas), por lo que una colección (en MySQL se denominan tablas) pueden existir documentos con esquemas diferentes.

Permite **replicación** (duplicación de datos), **maestro/esclavo** y **sharding** (particionado de datos entre diferentes servidor)

A continuación se puede ver un ejemplo de dos documentos de la misma colección:

```
{
Nombre: “Juan”,
Apellido: “Pérez”,
Curso: “2º ASIR”
}
```

```
{
Nombre: “José Cristian”,
Apellidos: “Cañaveras Vélez”,
Curso: “2º ASIR”,
Asignaturas: [
        {
        Nombre: “Servicios de Red e Internet”
        },
        {
        Nombre: “Administración de Sistemas Operativos”,
        },
        {
        Nombre: “Implantación de Aplicaciones Web”,
        }
]
```

**}**

Las consultas se realzan usando JavaScipt, algunas de las consultas mas importantes son las siguientes:

- **Create**

**db.coleccion.insert(documento)**

- **Read**

**db.coleccion.find/findOne(expresion)**

- **Update**

**db.coleccion.update(documento_original, documento_modificado)**

- **Delete**

**db.coleccion.remove(expresion)**

Para ello dispondremos de un servidor que tendrá un panel (clusterCP) que será el encargado de comprobar la conexión y el estado de los clusters.

Instalaremos Apache para el servidor web en el servidor MONITOR, MongoDB para la BBDD de autenticación de usuarios y registro de caídas, y PHP (además de la extensión ssh2). El servidor tendrá instalado Debian 8.

Instalamos primero apache.

```
# apt-get install apache2
```

Posteriormente instalaremos la librería SSH2 y el compilador de C build-essential.

```
# apt-get install php5-common libapache2-mod-php5 php5-cli
```

Tambien instalamos SSH2 para PHP, para ello primero descargamos la librería SSH2, la descomprimos y la compilamos.

```
#apt-get install build-essential
#apt-get install libssl-dev
#wget http://libssh2.org/download/libssh2-1.5.0.tar.gz
#tar vxzf libssh2-1.5.0.tar.gz
#cd libssh2-1.5.0
#./configure
#make
#make install
```

Instalamos y compilamos la librería php-ss2.

```
#pecl install ssh2 channel://pecl.php.net/ssh2-0.11.3
```

Ahora editamos el archivo **/etc/php5/apache/php.ini** y añadimos el modulo en la sección **Dinamic Extensions**.

```
extension=ssh2.so
```

Reiniciamos apache

```
#service apache2 restart
```

Ahora instalaremos mongoDB, mongoDB es una base datos documental, es decir, los datos no se guardan en tablas (filas y columnas) si no en documentos cuya agrupación se denomina colecciones, por lo que una colección en mongoDB se consideraría una tabla en MySQL como concepto de colección y los documentos se considerarían registros (tuplas). Por este motivo mongoDB no sigue un esquema fijo como en MySQL por lo que un documento puede tener campos que no estén disponibles en otros documentos, lo que permite más rapidez y un mejor rendimiento respecto a MySQL.

```
#apt-key adv --keyserver keyserver.ubuntu.com --recv 7F0CEB10
#echo 'deb http://downloads-distro.mongodb.org/repo/debian-sysvinit dist 10gen' | sudo tee /etc/apt/sources.list.d/mongodb.list
#apt-get update
#apt-get install -o apt::architecture=amd64 mongodb-10gen
```

Una vez instalado mongoDB reiniciamos el servicio **#service mongodb restart**

Una vez instalado crearemos una BBDD denominada **clustercp**, para ello ejecutaremos la consola de mongoDB.

**#mongo**

Una vez dentro de la consola crearemos la BBDD.

**#use clustercp**

NOTA: Con este comando accedemos a una base de datos existente y si no existe la crea.

Ahora crearemos las siguientes colecciones:

- **usuarios**: contendrá las credenciales para el inicio de sesión al sistema **clustercp**.

  Tendrá el siguiente esquema:

  - **usuario**
  - **password**

- **caídas**: guarda las caídas de cada nodo o nodos guardando el nombre del nodo, la fecha y la hora.

  Tendrá el siguiente esquema:

  - **nodo**
  - **fecha**
  - **hora**

- **clusters**: guardara los datos de los clusters, guardara el nombre, la dirección IP y los nodos que lo componen.

  Tendrá el siguiente esquema:

  - **nombre**
  - **ip**
  - **nodos**

- **balanceador**: guardara los datos de los balanceadores, guardara el nombre, la dirección IP.

  Tendrá el siguiente esquema:

  - **nombre**
  - **ip**

Para crearla usaremos el comando:

**db.createCollection(”nombreColeccion”)**

**db.createCollection(“usuarios”)**

**db.createCollection(“caidas”)**

**db.createCollection(“clusters”)**

**db.createCollection(“balanceadores”)**

Ahora le daremos seguridad para ello crearemos un usuario para la BBDD creada, el usuario será **jose** y la contraseña será **inves**.

**db.createUser({user:”jose”,pwd:”inves”})**

Una vez creadas las coleccion podremos verlas ejecutando:

**show collections**


***Imagen 10.1.2. Colecciónes de MongoDB.***

NOTA: Para autenticar el usuario usaremos:

**db.auth("usuario","contraseña")**

## 10.2 Inicio de sesión

Para acceder al sistema del clúster accederemos a la web **http://clustercp.josecristian.net** y deberemos loguearnos a través de un login con usuario y contraseña, dichas credenciales estarán guardadas en la BBDD **clustercp** y en la colección **usuarios**, la contraseña estará guardada con encriptación **MD5.**

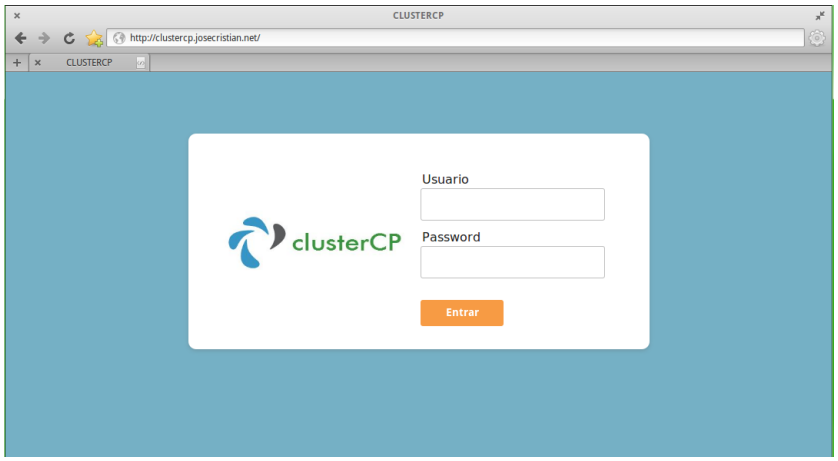


***Imagen 10.2.1. Login de clusterCP.***

Para el logueo comprobamos la colección **usuarios** de la BBDD **clustercp**, esta comprobación se realiza en el archivo **acceso.php**, para ello realizamos una consulta con el usuario y la contraseña pasada desde el formulario a través del metodo **POST** por seguridad, para realizar la consulta deberemos crear un array con el usuario y la contraseña pasadas y realizar la consulta con la propiedad **findOne** de mongoDB (ya que para acceder a mongoDB desde PHP deberemos usar **P**rogramacion **O**rientada a **O**bjetos).

Una vez enviado el array deberemos comprobar si se ha realizado bien la consulta, en dicho caso estableceremos una clave de sesión "**autentica**" con el valor **SI** para posteriormente comprobar que el usuario esta autenticado para que en caso de que el usuario acceda a un directorio sin estar logueado se le redireccione a la página de inicio de sesión. Posteriormente guardaremos la fecha y hora del acceso en la variable de sesión **"ultimoAccesso"** (lo usaremos en el archivo **seguridad.php**) y guardaremos el nombre de usuario en la variable de sesión **"usuario**", finalmente redirigiremos al usuario al directorio ./**dashboard/cluster**, si la consulta falla (el usuario o contraseña no son correctos) redirigeremos el usuario a la página de inicio y le pasaremos por **GET** el parámetro **errorusuario=error** el cual permitirá mostrar un mensaje de inicio de sesión erróneo.

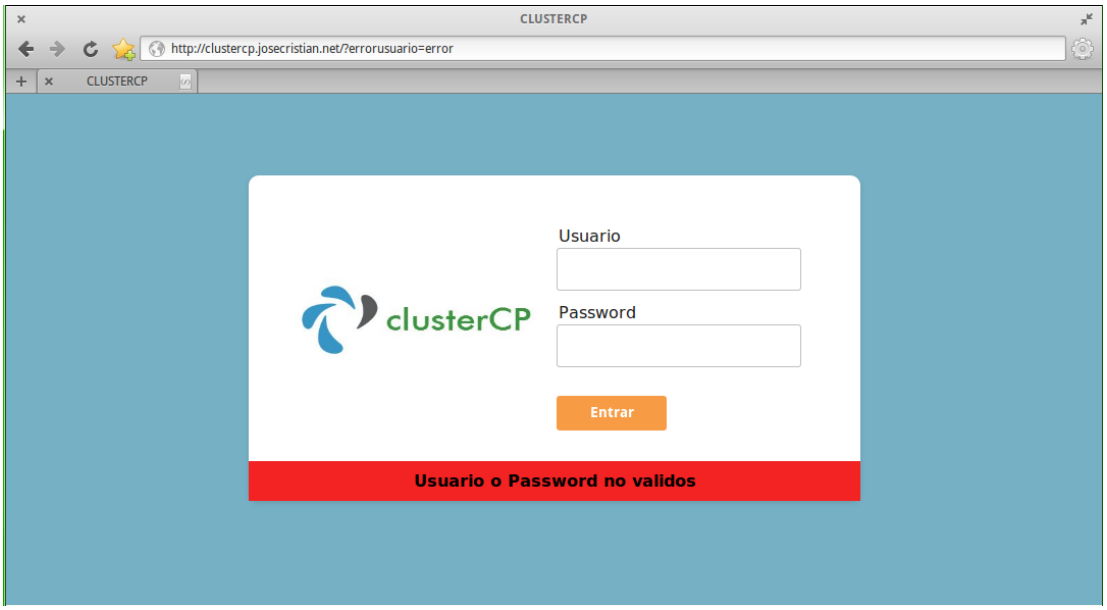


***Imagen 10.2.2. Mensaje de error de inicio de sesión.***

El código del archivo **acceso.php** es el siguiente.

```
<?php
// incluimos las credenciales de acceso a la BBDD clustercp
include("../app/db/conexion.php");
// accedemos a la coleccion usuario de la BBDD clustercp llamada $db
$coleccion = $db->usuarios;
// recogemos el usuario
$usuario=$_POST['usuario'];
// recogemos el password y lo convertimos a MD5 para comprobarlo con la BBDD
$pass=md5($_POST['password']);
// creamos el array
$consulta=array("usuario"=>$usuario,"password"=>$pass);
// realizaremos la consulta
```

```
        $user=$coleccion->findOne($consulta);
                // si la consulta es correcta entramos
                if ($user){
                        // iniciamos la sesion
                        session_start();
                        // autenticamos al usuario
                        $_SESSION["autentica"] = "SI";
                        // guardamos la fecha y hora del acceso
                        $_SESSION["ultimoAcceso"]= date("Y-n-j H:i:s");
                        // guaradamos el usuario
                        $_SESSION["usuario"]=$usuario;
                        // redireccionamos
                        header("Location: ./dashboard/estado_cluster");
                }else{
                        // redireccionamos al login con un mensaje de error
                        header("Location: ./?errorusuario=error");
                }
?>
```

El código del archivo **conexión.php** es el siguiente, en este archivo seleccionaremos la BBDD y nos autenticaremos, añadiremos este archivo en cada archivo PHP que realice alguna conexión a la BBDD (include).

```
<?php
        // instanciamos el objeto $conexion de la clase Mongo()
        $conexion = new Mongo();
        // accedemos a la propiedad del objeto conexion y le asignamos la
        BBDD clustercp
        $db = $conexion->clustercp;
        // autenticamos con el usuario y contraseña de la BBDD
        $db->authenticate('jose', 'inves');
?>
```

El código del archivo **seguridad.php** es el siguiente, en este archivo se comprueba si el usuario esta logueado a través de la variable de sesión “autentica” y comprobamos que el tiempo de cuando nos logueamos y el de ahora no es mayor a 10 minutos.

```
<?php
 //Reanudamos la sesión
 session_start();
 //Validamos si existe realmente una sesión activa o no
  if($_SESSION["autentica"] != "SI"){
  //Si no hay sesión activa, lo direccionamos al inicio de sesion
    header("Location: ../");
    exit();
  }else {
     //sino, calculamos el tiempo transcurrido
    $fechaGuardada = $_SESSION["ultimoAcceso"];
```

```
    $ahora = date("Y-n-j H:i:s");
    $tiempo_transcurrido = (strtotime($ahora)-strtotime($fechaGuardada));
    //comparamos el tiempo transcurrido
      if($tiempo_transcurrido >= 600) {
        //si pasaron 10 minutos o más destruimos la sesion
        session_destroy();
        //enviamos al usuario al login
        header("Location: ../");
      }else {
        //sino, actualizamos la fecha de la sesión
        $_SESSION["ultimoAcceso"] = $ahora;
      }
   }
?>
```

Dispondremos en un fichero **salir.php** que accederemos a el desde un botón “SALIR” cuya función será destruir la sesión del usuario y redireccionarlo a la página de login.

```
<?php
        session_start();
        // destruimos la sesion
        session_destroy();
        // redireccionamos al login
        header("Location: ./");
?>
```

## 10.3 Monitorización de Clúster

En esta sección dispondremos un una imagen de servidor por cada nodo con un icono de ok o de error, la página se actualizará cada minuto. A través de esta página en primer lugar realizamos un ping a la IP de cada clúster obtenida de la BBDD, si no se realiza ping se extrae de la colección clusters los nodos de cada clúster y se “pintan” en la página con el error, si se realiza ping se realiza la conexión SSH al servidor, si no se accede a través de SSH se extrae de la BBDD los nodos y se “pintan” los nodos con el error, si accedemos vía SSH comprobamos los nodos online y los nodos offline a través de comando **pcs status nodes** (utilizando el comando **tail** y **head** extraemos los nodos de cada línea) y filtramos los resultado obtenido para obtener el nombre del nodo.

El código PHP está preparado para “pintar” n clusters, ya que se realiza un foreach de cada clúster que está en la colección.

He de aclarar que los nodos que están en la colección **clusters** únicamente se accede a la colección en caso de fallo del ping o de conexión SSH, ya que si todo está correctamente el nombre de los nodos se extraen a

través de la conexión SSH del servidor, por lo que de la colección **clusters** en caso de que no haya error extraeremos la dirección IP y el nombre del clúster.

Si todo está correctamente.

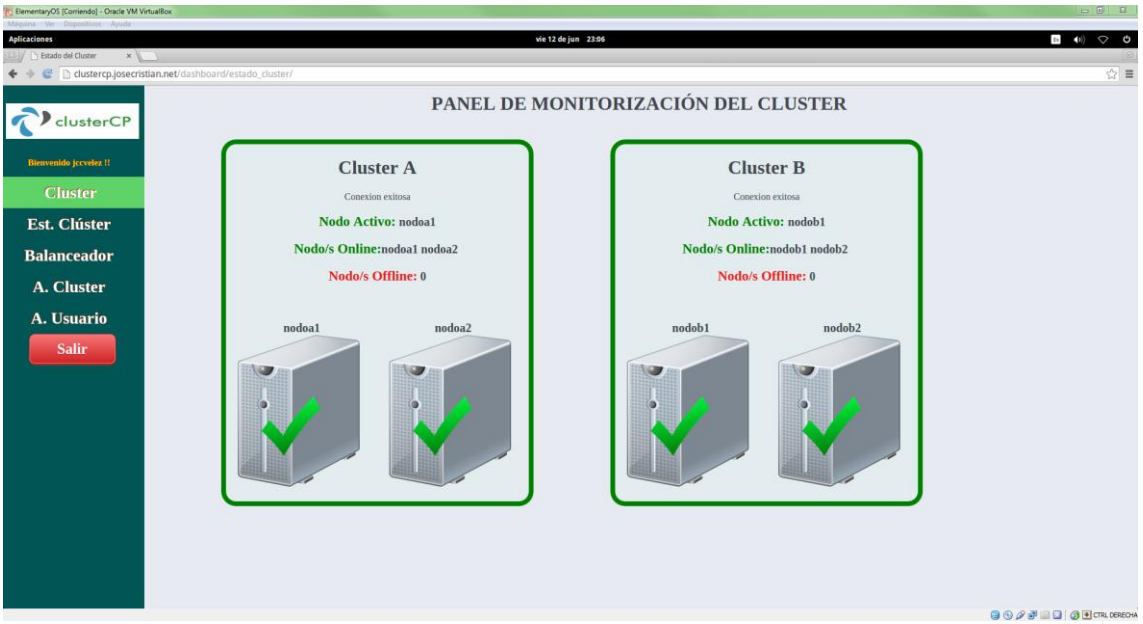


***Imagen 10.3.1. Estado correcto de los nodos.***

A continuación se puede ver si uno de los nodos falla.

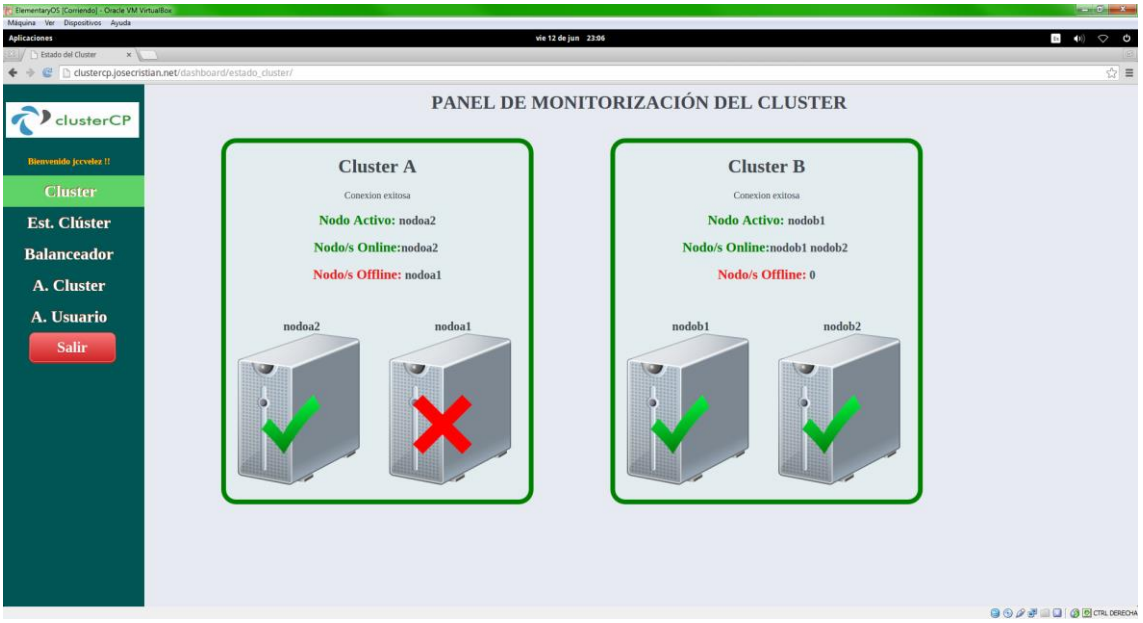


***Imagen 10.3.2. Nodo fallido.***

Si un clúster completo falla.

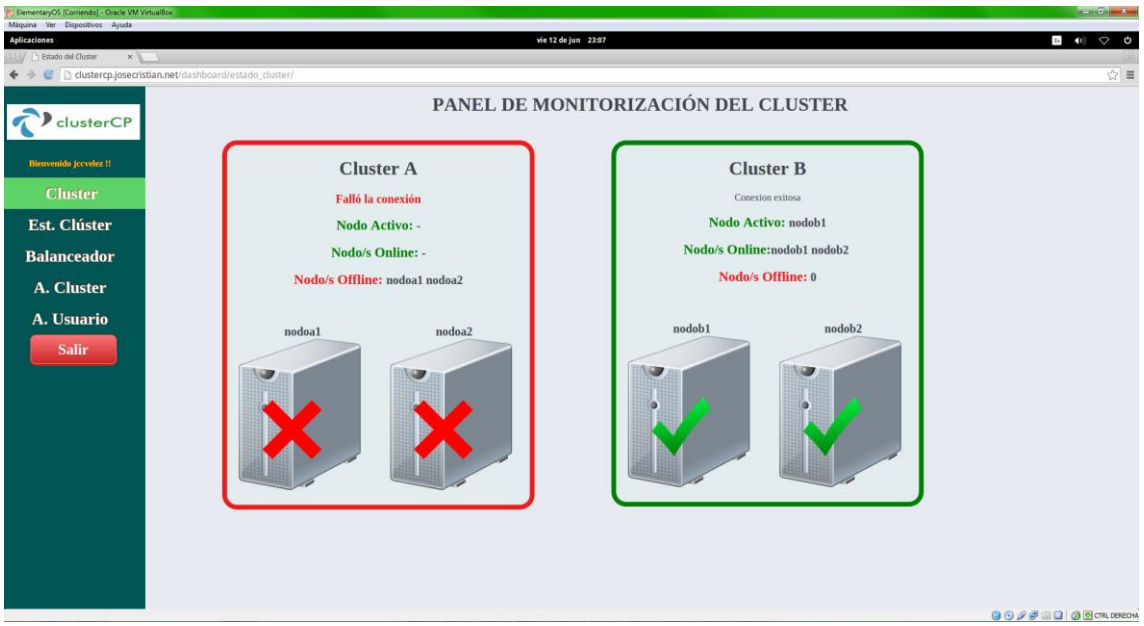


***Imagen 10.3.3. Cluster fallido completo.***

Cuando un clúster **completo**, dado que no se puede acceder al clúster a través de SSH se extraerán los nodos que componen el clúster desde la colección **clusters** de la BBDD clustercp, el borde de los nodos será de color **rojo**.

El código del archivo de inicio **/web/dashboard/estado_cluster/index.php** es el siguiente.

```
<?php
   // nombre de la pagina para la opcion del menu
   $pagina="estado_cluster";
   // Conexion con el controlador
   include("../../../app/controladores/estado_cluster_controlador.php");
?>
```

El código del controlador **/app/controladores/estado_cluster_controlador.php** que “une” la vista con el modelo es el siguiente.

```
<?php
   // nombre de la pagina para la opcion del menu
   $pagina="estado_cluster";
   // Conexion con el controlador
   include("../../../app/controladores/estado_cluster_controlador.php");
?>
```

El código del modelo /**app/modelos/estado_cluster_modelo.php** que es el encargado de extraer los datos de la BBDD y generar los datos.

```
<?php
        // quitamos en produccion los posibles warning
    error_reporting(0);
    // inclumos el archivo de seguridad
    include("../../../app/includes/seguridad.php");
    // inclumos el archivo que tendra los datos de los clusters
    // de la BBDD
    include_once("../../../app/includes/ssh.php");
?>
```

Para la extracción de los datos a través de la colección **clusters** se crear un array por valor, es decir, un array con los nombres, otro para las direcciones IP, y otro para los nodos (si el clúster está completamente caido). Posteriormente se recorre cada a la vez la misma columna de cada array a través de un índice.

Es decir, tenemos los siguientes arrays:

$arraynombre(‘Cluster A’,’Cluster B’);
$arrayip(‘192.168.50.254’,’192.168.50.253’);
$arraynodos(‘nodoa1 nodoa2’,’nodob1 nodob2’);

Posteriormente recorremos los arrays, la primera vez extraeremos la primera posición de cada array extrayendo en cada recorrido los datos del mismo cluster.

Primer recorrido: Cluster A, 192.168.50.254, nodoa1 nodoa2
Segundo recorrido: Cluster B, 192.168.50.253, nodob1 nodob2

El código de la vista **/app/vistas/estado_cluster_vista.php** que muestra el contenido al usuario, es decir, el estilo de los datos.

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="refresh" content="60">
        <title>Estado del Cluster</title>
        <!-- estilo del menu y fondo-->
        <link rel="styleSheet" href="../css/general.css" type="text/css">
        <!-- Estilo del cluster-->
        <link rel="styleSheet" href="./css/estilo.css" type="text/css">
    </head>
    <body>
        <div id="padre">
            <!-- Incluimos el menu -->
```

```
        <?php include_once("../includes/sidebar.php"); ?>
        <div id="contenido">
          <div class="cabecera"><h1>PANEL DE MONITORIZACIÓN DEL
CLUSTER</h1></div>
            <?php
              // realizamos un bucle con cada una de las direcciones IP de
cada cluster, la primera
              // de cada array columna se denominara columna, para
acceder a la misma posicion
              foreach ($arrayip as $columna => $ipcluster) {
                // realizamos un ping a la IP y recogemos el resultado
              $ping=exec("ping -c 1 -w 1 ".$ipcluster, $input, $resultado);
              // creamos un array con los nodos obtenidos de MongoDB
              $arraynodosextraido=explode(" ", $arraynodos[$columna]);
                // si el ping es exitoso
                if ($resultado==0) {
                  // realizamos la conexion SSH al servidor
                  if(!($con = ssh2_connect($ipcluster, 22))){
                    // si no nos conectamos pintamos el cluster con el
error,
                    // los nodos obtenidos de ese cluster de la BBDD, el
nombre del cluster.
                    echo "<div class='caja
bordeestadoofflineall'><h1>".$arraynombre[$columna]."</h1>
                    <p class='falloconexiontitulo'>El clúster esta caido</p>
                    <p><span class='minititulo online'>Nodo Activo:
</span>-</br></p>
                    <p><span class='minititulo online'>Nodo/s Online:
</span>-</br></p>
                    <p><span class='minititulo offline'>Nodo/s Offline:
</span>".$arraynodos[$columna]."</p>
                    <div>";
                    // pintamos las imagenes de los nodos con el nombre
de nodo y el error.
                    foreach ($arraynodosextraido as $arraynodo) {
                      echo "<p
class='server'><br>".$arraynodo."</br><img src='./img/error.png' alt="
class='error'><img src='./img/server.png' alt='servidor1'></p>";
                    }

                    echo "</div>
                    </div>
                    ";
                    // si realizamos la conexion a traves de SSH
                  } else {
                    // si el usuario o contraseña son erroneos mostramos
un mensaje
                    if(!ssh2_auth_password($con, "root", "inves")) {
                      echo "Imposible autenticar";
                    // si realizamos la conexion
```

```
                        } else {
                            // mostramos el nombre del cluster obtenido del
array $arraynombre
                            // y de la misma posicion de la direccion IP estraida
                            echo "<div class='caja
bordeestadoonline'><h1>".$arraynombre[$columna]."</h1>Conexion
exitosa</br>";
                            //NODOS ONLINE
                               //ejecutamos el comando para extraer lod nodos
online
                               if (!($datos = ssh2_exec($con, "pcs status nodes |
head -2 | tail -1"  ))) {
                                  // mostramos un error
                                   echo "fallo, no se puede enviar el comando";
                               } else {
                               // si el comando se ejecuto correctamente
                                  // recogemos los datos que nos devolvio el
comando
                                  stream_set_blocking($datos, true);
                                  // establecemos la variable $nodosonline
                                  $nodosonline = "";
                                     while ($buf = fread($datos,4096)) {
                                        $nodosonline .= $buf;
                                     }
                                  // filtrado de resultados
                                     // quitamos el Online:
                                  $nodosonline=str_replace("Online:
","",$nodosonline);
                                  //creamos el array sin espacios al inicio y final
                                  $nodosonline=explode(' ', trim($nodosonline));
                                  // mostramos el nodo activo, el primer nodo es
el activo
                                  echo "<p><span class='minititulo online'>Nodo
Activo: </span>".$nodosonline[0]."</br></p>";
                                  // mostramos los nodos online
                                  echo  "<p><span class='minititulo
online'>Nodo/s Online:</span>";
                                  foreach ($nodosonline as $nodoson) {
                                     echo $nodoson." ";
                                  }
                                  echo "</br></p>" ;



                                  //NODOS OFFLINE
                                     //extraemos los nodos offline a traves del
comando
                                  $datos = ssh2_exec($con, "pcs status nodes |
head -4 | tail -1");
```

```
                                stream_set_blocking($datos, true);
                                // establecemos la variable
                                $nodosoffline = "";
                                   while ($buf = fread($datos,4096)) {
                                      $nodosoffline .= $buf;
                                   }

                                // filtrado de resultados
                                   // quitamos el valor Offline:
                                   $nodosoffline=str_replace("Offline:
","",$nodosoffline);
                                   // quitamos los espacios al inicio y al fin para
evitar errores
                                   $nodosoffline=trim($nodosoffline);

                                   if ($nodosoffline){
                                      // si hay nodos offline creamos un array con
ellos y quitamos los espacios
                                      $nodosoffline=explode('
',trim($nodosoffline));
                                      // mostramos los nodos offline desde el
array
                                      echo  "<p><span class='minititulo
offline'>Nodo/s Offline:</span> ";
                                       foreach($nodosoffline as $nodooff) {echo
$nodooff." ";}
                                       echo "</p>";

                                   }else{
                                      // si no hay ningun nodo offline mostramos
0
                                      echo  "<p><span class='minititulo
offline'>Nodo/s Offline:</span> 0</p>" ;

                                   }

                                   //pintamos las imagenes de cada nodo online
desde el array obtenido

                                   foreach ($nodosonline as $nodoonline) {
                                      echo "<p
class='server'><br>".$nodoonline."</br><img src='./img/ok.png' alt="
class='ok'><img src='./img/server.png' alt='servidor1'></p>";

                                   }

                                   //pintamos las imagenes de cada nodo offline
desde el array obtenido

                                   if($nodosoffline){
```

```
                                    foreach ($nodosoffline as $nodooffline) {
                                        echo "<p
class='server'><br>".$nodooffline."</br><img src='./img/error.png' alt="
class='error'><img src='./img/server.png' alt='servidor1'></p>";
                                    }
                                }

                                //cerramos la conexion SSH
                                fclose($datos);
                            }
                            echo "</div>";
                        }
                    }
                }else{
                    // si no realizamos ping mostramos el cluster con error y
el nombre y nodos obtenido de la BBDD
                    // a traves de la posicion sacamos el nombre del cluster y
el nombre del nodo de cada array.
                    echo "<div class='caja
bordeestadoofflineall'><h1>".$arraynombre[$columna]."</h1>
                    <p class='falloconexiontitulo'>Falló la conexión</p>
                    <p><span class='minititulo online'>Nodo Activo: </span>-
</br></p>
                    <p><span class='minititulo online'>Nodo/s Online:
</span>-</br></p>
                    <p><span class='minititulo offline'>Nodo/s Offline:
</span>".$arraynodos[$columna]."</p>
                    <div>";
                    foreach ($arraynodosextraido as $arraynodo) {
                        echo "<p
class='server'><br>".$arraynodo."</br><img src='./img/error.png' alt="
class='error'><img src='./img/server.png' alt='servidor1'></p>";
                    }
                    echo "</div>
                    </div>
                    ";
                }
}
            ?>
        </div>
        <br style="clear:both"/>
    </div>
  </body>
</html>
```

## 10.4 Estadísticas del Clúster

A través de esta página podremos visualizar las caídas de cada nodo, cada nodo que falla se guarda en la BBDD, cada 60 segundos se comprueba el estado de los nodos, si uno de ellos falla se guarda un nuevo documento en la colección **caídas**, esto se realiza a través de una tarea programada que ejecutara el código PHP a través de CURL que registrara en la colección el nodo que presenta el error, la fecha y la hora (el código que registra estos errores esta explicado en el punto **9.9**).

Para acceder a la página de estadísticas pulsaremos la opción **Est. Cluster** en el menú.


***Imagen 10.4.1. Menú.***

Una vez dentro podremos ver las estadísticas de cada nodo, el número de caídas por fecha y las caídas por nodo, también podremos ver la RAM usada por cada servidor expresado en %.

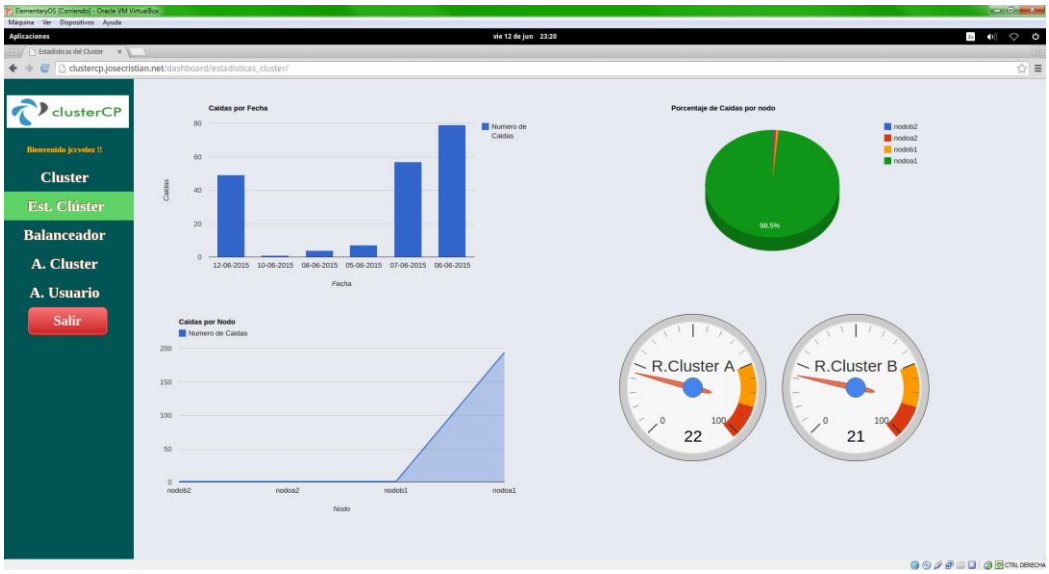


***Imagen 10.4.2. Panel de estadísticas.***

Para mostrar las estadísticas realizamos las siguientes consultas a la colección **caídas:**

- **Caídas por fecha:** contamos ($sum) los nodos agrupados por fecha realizando una agregación.
- **Caídas por nodo:** contamos los documentos agrupados por nodo, es decir agrupamos las caídas por nodo.
- **Porcentaje de caídas por nodo:** Usaremos la consulta anterior.

Para mostrar la memoria RAM extraeremos los datos usando el comando **free –m** a través de SSH a cada clúster (al nodo activo de cada clúster), realizamos la conexión utilizando la dirección IP extraída de la colección **clusters** y buscaremos la palabra **Mem:** y extraemos la tercera columna (**awk {'print $3'}**).

Una vez extraídos los datos **"pintaremos"** los gráficos usando la librería **CHART** de **GOOGLE**, mostrando los datos por columnas, los gráficos se extraen a través de JAVASCRIPT.

El código del archivo de inicio **/web/dashboard/estadisticas_cluster/index.php** es el siguiente.

```
<?php
  // nombre de la pagina para el menu
  $pagina='estadisticas_cluster';
  // Conexion con el controladors
  include("../../../app/controladores/estadisticas_cluster_controlador.php");
?>
```

El código del controlador **/app/controladores/estadisticas_cluster_controlador.php** que "une" la vista con el modelo es el siguiente.

```
<?php
        // incluimos el modelo
        include("../../../app/modelos/estadisticas_cluster_modelo.php");
        // incluimos la vista
        include("../../../app/vistas/estadisticas_cluster_vista.php")
?>
```

El código del modelo /**app/modelos/estadisticas_cluster_modelo.php** que es el encargado de extraer los datos de la BBDD y generar los datos.

```
<?php
  // quitamos en produccion los posibles warning
  error_reporting(0);
  // conexion con la BBDD clustercp
  include('../../../app/db/conexion.php');
  // direccion IP, nombre de nodos de la BBDD, y nombre del cluster
```

```
 include("../../../app/includes/ssh.php");
 // accedemos a la coleccion caidas
 $coleccion=$db->caidas;

 //Estadistica columnas
  // agrupamos los nodos (contamos los nodos) por fecha
 $resultadofechacaidas=  $coleccion->aggregate(array('$group' =>
array('_id' => array('fecha'=>'$fecha'), 'caidas' => array('$sum' => 1))));

 //Estadistica donus/curva
  // agrupamos las caidas por nodo
 $resultadonumcaidas=  $coleccion->aggregate(array('$group' =>
array('_id'=> array('nodo'=> '$nodo'), 'caidas' => array('$sum' => 1))));

 // indicador RAM

 ////////////  Conexion de cada cluster para extraer la RAM   ///////////
  // establecemos el array para la memoria usada de cada cluster
 $arraymemusada=array();
  // comprobamos la ram de cada IP almacenada en la coleccion
 foreach ($arrayip as $ip) {

  // nos conectamos con el servidor por SSH
  if(!($con = ssh2_connect($ip, 22))){

  } else {
     // autenticamos
     if(!ssh2_auth_password($con, "root", "inves")) {
        echo "Imposible autenticar";
     } else {

        // ejecutamos el comando y extraemos
        // la tercera columna que muestra la RAM
        if (!($datos = ssh2_exec($con, "free -m | grep 'Mem:' | awk {'print
$3'}"  ))) {
           echo "fallo, no se puede enviar el comando";
        } else {
           // recojemos los datos del comando
           stream_set_blocking($datos, true);
           // establecemos la variable que guardara la memoria usada
           $memoriausadaa = "";
              while ($buf = fread($datos,4096)) {
                 $memoriausadaa .= $buf;
              }

           // Extraemos la memoria RAM total que esta
           // en la segunda columna para poder hacer el porcentaje
           if (!($datos = ssh2_exec($con, "free -m | grep 'Mem:' | awk {'print
$2'}"  ))) {
                    echo "fallo, no se puede enviar el comando";
```

```
                    } else {
                      // recogemos los datos
                      stream_set_blocking($datos, true);
                      $memoriatotala = "";
                        while ($buf = fread($datos,4096)) {
                          $memoriatotala .= $buf;
                        }
                      // realizamos el porcentaje multiplicando la memoria
                      // usada por 100 y dividiendola con la memoria total,
                      // la redondeamos quitando los decimales
                      $memoriausadaa=
round(($memoriausadaa*100)/$memoriatotala,0)." %";
                      // volvemos el valor a entero para evitar problemas
                      $memoriausadaa=(int)$memoriausadaa;
                      // añadimos al array la memoria usada actual al array
anterior,
                      // vamos arrastrando los valores de ram de cada cluster
                      $memoriausadaa=array_push($arraymemusada,
$memoriausadaa);
                    }

          }
              // cerramos la conexion
              fclose($datos);
        }
    }
  }

?>
```

El código de la vista **/app/vistas/estadisticas_cluster_vista.php** que muestra el contenido al usuario, es decir, el estilo de los datos.

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="refresh" content="60">
   <title>Estadisticas del Cluster</title>
   <!-- estilo del menu y fondo-->
   <link rel="styleSheet" href="../css/general.css" type="text/css">
   <!-- Estilo y posicion de las cajas de estadisticas-->
   <link rel="styleSheet" href="./css/estadisticas.css" type="text/css">
   <!-- Libreria CHART Google -->
   <script type="text/javascript"
src="https://www.google.com/jsapi"></script>
   <!--Estadistica columnas-->
        <script type="text/javascript">
     google.load("visualization", "1", {packages:["corechart"]});
```

```
    google.setOnLoadCallback(drawVisualization);

    function drawVisualization() {
      var data = google.visualization.arrayToDataTable([
        ['Fecha', 'Numero de Caidas'],
        // extraemos la fecha de cada nodo a traves del array
multidimensional
        // y el numero de caidas
        <?php
          foreach ($resultadofechacaidas as $valor) {
            foreach ($valor as $valor2) {
              echo "['".$valor2['_id']['fecha']."',".$valor2['caidas']."],";
            }
          }

        ?>
    ]);

   var options = {
     title : 'Caidas por Fecha',
     backgroundColor: '#E7EBF1',
     vAxis: {title: "Caidas"},
     hAxis: {title: "Fecha"},
     seriesType: "bars",
     series: {5: {type: "line"}}
   };

   var chart = new
google.visualization.ComboChart(document.getElementById('columnas'));
   chart.draw(data, options);
    }
   </script>

  <!--Estadistica donus-->
        <script type="text/javascript">
   google.load("visualization", "1", {packages:["corechart"]});
   google.setOnLoadCallback(drawChart);
   function drawChart() {
     var data = google.visualization.arrayToDataTable([
       ['Nodo', 'Numero de Caidas'],
       // extraemos el nombre de cada nodo a traves del array
multidimensional
       // y el numero de caidas
         <?php
           foreach ($resultadonumcaidas as $valor) {
             foreach ($valor as $valor2) {
               echo "['".$valor2['_id']['nodo']."',".$valor2['caidas']."],";
             }
           }
         ?>
```

```
      ]);
      var options = {
        title: 'Porcentaje de Caidas por nodo',
        is3D:true,
        backgroundColor: '#E7EBF1',
        chartArea:{width:'80%'},
      };

      var chart = new
google.visualization.PieChart(document.getElementById('donus'));
      chart.draw(data, options);
    }
  </script>
    <!--Estadistica curva-->
  <script type="text/javascript">
    google.load("visualization", "1", {packages:["corechart"]});
    google.setOnLoadCallback(drawChart);
      function drawChart() {
        var data = google.visualization.arrayToDataTable([
          ['Nodo', 'Numero de Caidas'],
          // extraemos el nombre de cada nodo a traves del array
multidimensional
          // y el numero de caidas
          <?php
            foreach ($resultadonumcaidas as $valor) {
              foreach ($valor as $valor2) {
                echo "['".$valor2['_id']['nodo']."',".$valor2['caidas']."],";
              }
            }
          ?>
        ]);

        var options = {
          title: 'Caidas por Nodo',
          hAxis: {title: 'Nodo',  titleTextStyle: {color: '#333'}},
          vAxis: {minValue: 0},
          isStacked: 'relative',
          legend: {position: 'top', maxLines: 3},
          backgroundColor: '#E7EBF1',
          chartArea:{width:'80%'}
        };
        var chart = new
google.visualization.AreaChart(document.getElementById('area'));
        chart.draw(data, options);
      }
  </script>

          <!--Estadistica indicador clustera-->
  <script type="text/javascript">
    google.load("visualization", "1", {packages:["gauge"]});
```

```
    google.setOnLoadCallback(drawChart);
    function drawChart() {
      var data = google.visualization.arrayToDataTable([
        // extraemos el nombre de cada nodo y su memoria RAM usada
        ['Nodo', 'Memoria'],
        <?php
          foreach ($arraynombre as $columna => $cluster) {
            echo  "['R.".$cluster."', ".$arraymemusada[$columna]."],";
           }
        ?>
      ]);
      var options = {
        width: 600, height: 600,
        backgroundColor: '#E7EBF1',
        redFrom: 90, redTo: 200,
        yellowFrom:75, yellowTo: 90,
        minorTicks: 5,
      };
      var chart = new
google.visualization.Gauge(document.getElementById('indicador'));
      chart.draw(data, options);
    }
    </script>

  </head>
  <body>
  <div id="padre">
    <?php include_once("../includes/sidebar.php") ?>
    <div id="contenido">
            <div class="estadisticas">
        <div id="columnas" style="width: 750px; height: 400px"
class="cajas"></div>
        <div id="donus" style="width: 750px; height: 400px;"
class="cajas"></div>
        <div id="area" style="width: 750px; height: 400px;"
class="cajas"></div>
        <div id="indicador" style="width: 750px; height: 400px;" class="cajas
indicador"></div>
            </div>
    </div>
    <br style="clear:both"/>
  </div>
  </body>
</html>
```

## 10.5 Balanceador

El panel para la administración del balanceador de carga permite cambiar el tipo de algorimo, podemos elegir entre los siguiente algorimos:

- Roundrobin
- Leastconn
- Source

El panel muestra el algoritmo que usan los balanceadores en el momento (ya que ambos forman uno solo de vista al usuario), una vez que el usuario cambiar el algoritmo el cambio se realiza en todos los balanceadores disponibles, para ello se extraen las IPs de los balanceadores de la BBDD y se crear un array que posteriormete será recorrido realizando el cambio.

Cuando el usuario abre la página se muestra lo siguiente:


***Imagen 10.5.1. Formulario del cambio de algoritmo del balanceador de carga.***

Al elegir un nuevo algoritmo y pulsar “cambiar” se envía el comando del cambio y se actualiza la página con el nuevo algoritmo, tanto en la línea superior como en el “radio” marcado.

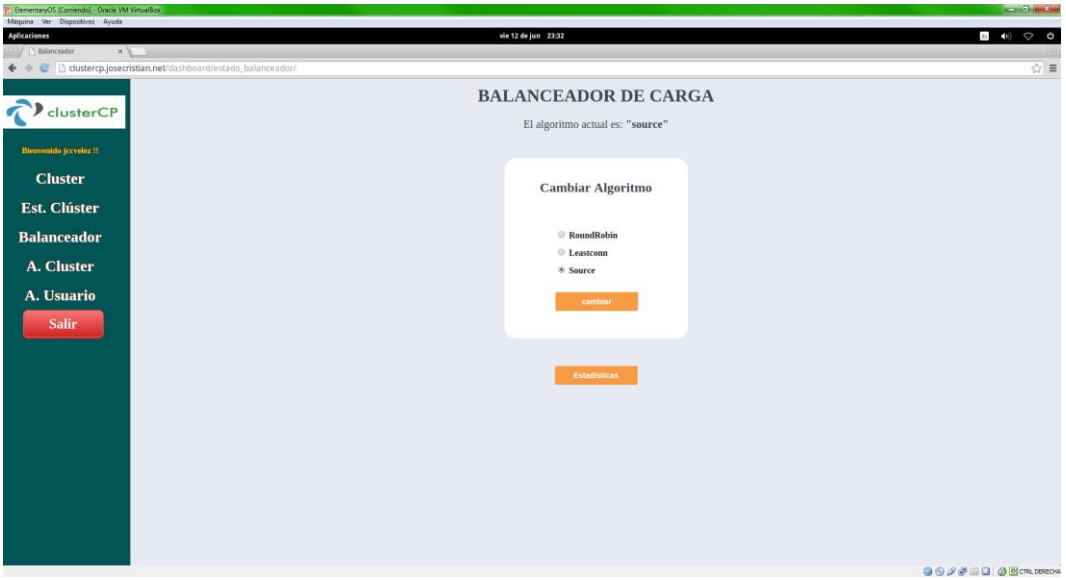


***Imagen 10.5.2. Cambio de algoritmo realizado.***

Para extraer el tipo de algoritmo se extrae la segunda columna de la línea que contiene la palabra “**balance**” que indica el algoritmo.

Para realizar el cambio se usa el comando **sed** que realiza una modificación del valor anterior (guardado en la variable $tipoantes) y se modifica por el tipo actual marcado **$tipoactual** en el archivo **haproxy.cfg**.

```
sed  -i  's/".trim($tipoantes)."/".$_POST['tipoahora']."/g' /etc/haproxy/haproxy.cfg
```

El código del archivo de inicio **/web/dashboard/estado_balanceador/index.php** es el siguiente.

```
<?php
  // nombre de la pagina para el menu
  $pagina='estado_balanceador';
  // Conexion con el controlador
  include("../../../app/controladores/estado_balanceador_controlador.php");
?>
```

El código del controlador **/app/controladores/estado_balanceador_controlador.php** que “une” la vista con el modelo es el siguiente.

```
<?php
        // incluimos el modelo
        include '../../../app/modelos/estado_balanceador_modelo.php';
        // incluimos la vista
        include '../../../app/vistas/estado_balanceador_vista.php';
?>
```

El código del modelo /**app/modelos/estado_balanceador_modelo.php** que es el encargado de extraer los datos de la BBDD y generar los datos.

```
<?php
   // quitamos en produccion los posibles warning
   error_reporting(0);
   // inclumos el archivo que tendra los datos de los clusters
   // de la BBDD
   include("../../../app/db/conexion.php");
   // nombre de la pagina para la opcion del menu
   $pagina="cluster";
   //seleccionamos la coleccion balanceadores
   $coleccion=$db->balanceadores;
   // definimos el array
   $arraybalanceadorip=array();
   // extraemos todos los registros
   $balanceadores=$coleccion->find();
   foreach ($balanceadores as $datosbalanceador) {
      foreach ($datosbalanceador as $columna=>$ipbalanceador) {
         if($columna=="ip") array_push($arraybalanceadorip,
$ipbalanceador);
      }
   }

   //extraer algoritmo actual
   function extraeralgoritmo($arraybalanceadorip){

      foreach ($arraybalanceadorip as $ip) {
         // Aqui abrimos la conexion SSH y en el tipo enviamos el comando.
         // si el usuario o contraseña son erroneos mostramos un mensaje
         if(!($con = ssh2_connect($ip, 22))){
            echo "Error de conexion SSH";
            // si realizamos la conexion a traves de SSH
         } else {
            if(!ssh2_auth_password($con, "root", "inves")) {
               echo "Imposible autenticar";
               // si realizamos la conexion
            } else {
               $comando="grep 'balance' /etc/haproxy/haproxy.cfg | awk
{'print $2'}";
               if (!($datos = ssh2_exec($con, $comando))) {
                  // mostramos un error
                  echo "fallo, no se puede enviar el comando";
               } else {
                  // recogemos los datos
                  stream_set_blocking($datos, true);
                  $tipoantes="";
                  while ($buf = fread($datos,4096)) {
                     $tipoantes .= $buf;
                  }
```

```
                    return $tipoantes;
                    //cerramos la conexion SSH
                    fclose($datos);
                }
            }
        }
    }
}
// llamamos a la funcion para extraer el tipo de algoritmo actual
$tipoantes= extraeralgoritmo($arraybalanceadorip);

if(isset($_POST['cambiar'])){
    /* extraemos la direcciones IP de los balanceadores
       de la BBDD a traves de un array y creamos un foreach
       para recorrer cada IP
    */
    foreach ($arraybalanceadorip as $ip) {
        // Aqui abrimos la conexion SSH y en el tipo enviamos el comando.
        // si el usuario o contraseña son erroneos mostramos un mensaje
        if(!($con = ssh2_connect($ip, 22))){
            echo "Error de conexion SSH";
            // si realizamos la conexion a traves de SSH
        } else {
            if(!ssh2_auth_password($con, "root", "inves")) {
                echo "Imposible autenticar";
                // si realizamos la conexion
            } else {
                // comprobamos que lo recibido es un valor de los permitidos
y evitar que se modique el html
                if($_POST['tipoahora']=="roundrobin" or
$_POST['tipoahora']=="leastconn" or $_POST['tipoahora']=="source") {
                    // generamos la el comando, ya que es compuesto
                    $comando="sed  -i
's/".trim($tipoantes)."/".$_POST['tipoahora']."/g'
/etc/haproxy/haproxy.cfg";
                     if (!($datos = ssh2_exec($con, $comando))) {
                        // mostramos un error
                        echo "fallo, no se puede enviar el comando";
                    } else {
                        // echo  Modo cambiado correctamente de $tipoantes a
$tipoahora
                        if (!($datos = ssh2_exec($con, "service haproxy restart")))
{
                            // mostramos un error
                            echo "fallo, no se puede enviar el comando";
                        }
                        //cerramos la conexion SSH
                        fclose($datos);
                    }
                }
```

```
                echo "</div>";
            }
        }
    }
    // llamamos a la funcion para extraer el tipo de algoritmo actual
    $tipoantes= extraeralgoritmo($arraybalanceadorip);
?>
```

El código de la vista **/app/vistas/estado_balanceador_vista.php** que muestra el contenido al usuario, es decir, el estilo de los datos.

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8">
        <title>Balanceador</title>
        <!-- estilo del menu y fondo-->
        <link rel="styleSheet" href="../css/general.css" type="text/css">
        <!-- Estilo del cluster-->
        <link rel="styleSheet" href="./css/estilo.css" type="text/css">
        <!-- Estilo del formulario-->
        <link rel="styleSheet" href="../css/form.css" type="text/css">
    </head>
    <body>
        <div id="padre">
            <!-- Incluimos el menu -->
            <?php include_once("../includes/sidebar.php"); ?>
            <div id="contenido" >
                <div class="cabecera centrado"><h1>BALANCEADOR DE
CARGA</h1></div>
                <div><p class="centrado tamanioletra">El algoritmo actual es:
<b>"<?php echo trim($tipoantes) ?><b>"</p></div>
                <div class="centrado">
                    <center>
                        <table>
                            <tbody>
                                <tr>
                                    <td style="padding: 20px 40px 0 40px;">
                                        <form method="post" action="">
                                            <table class="caja_reset">
                                                <center>
                                                    <h2>Cambiar Algoritmo</h2>
                                                </center>
                                                <tbody>
                                                    <tr>
                                                        <!-- Comprobamos el tipo de algoritmo
que hay actualmente para dejarlo seleccionado -->
```

```
                                    <td style="padding: 12px 0 0
2px;"><input type="radio" name="tipoahora" value="roundrobin" <?php if
(trim($tipoantes)=="roundrobin") echo "checked"; ?>> RoundRobin</td>
                                </tr>
                                <tr>
                                    <td style="padding: 12px 0 0
2px;"><input type="radio" name="tipoahora" value="leastconn" <?php if
(trim($tipoantes)=="leastconn") echo "checked"; ?>> Leastconn</td>
                                </tr>
                                <tr>
                                    <td style="padding: 12px 0 0
2px;"><input type="radio" name="tipoahora" value="source" <?php if
(trim($tipoantes)=="source") echo "checked"; ?>> Source </br><span
style="padding:0 0 0 14px;"></span></td>
                                </tr>
                                <tr>
                                    <td style="padding: 12px 0 0
2px;"><center><input type="submit" name="cambiar" value="cambiar"
class="boton_reset"></center></td>
                                </tr>

                            </tbody>
                        </table>
                    </form>
                </td>
            </tr>
        </tbody>
    </table>
    <button class="boton_reset"><a
href="http://clustercp.josecristian.net/haproxy?stats" target="_blank"
class="enlace">Estadísticas</a></button>

        </center>
    </div>
</div>
<br style="clear:both"/>
</div>
</body>
</html>
```

## 10.6 Añadir Clúster

Cuando añadimos un clúster a la colección **clusters** (nombre, ip, nodos) podemos visualizar en el menú **"cluster"** el nodo añadido, ya que automáticamente se extrae de la base de datos la dirección IP, nombre y nodos, los nodos únicamente se extraen si el clister completo esta caído ya que no se puede conectar por SSH.

El usuario puede agregar el clister a través del formulario.

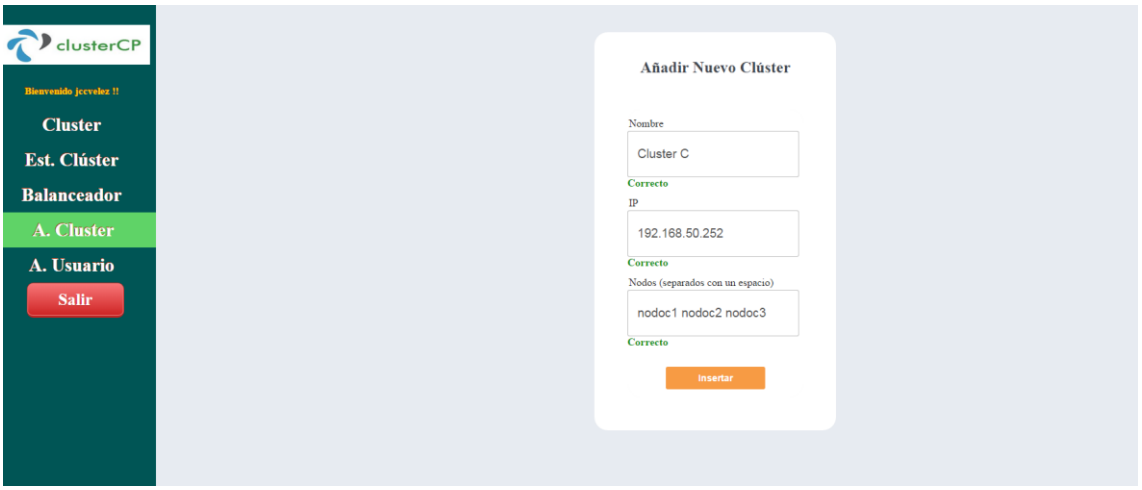


***Imagen 10.6.1. Creación de nuevo clúster.***

Si todo está correctamente se nos mostrará un mensaje en el que nos indicara de que se ha indicado correctamente.

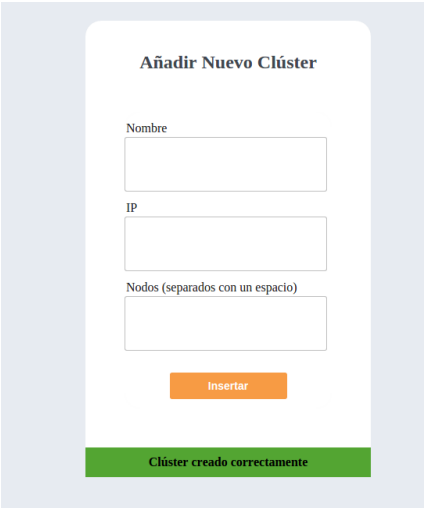


***Imagen 10.6.2. Cúster creado correctamente.***

Si accedemos al menú clúster veremos el nuevo clúster añadido, pero se mostrara caído ya que el clúster con la IP insertada no existe y no se puede conectar a través de SSH.

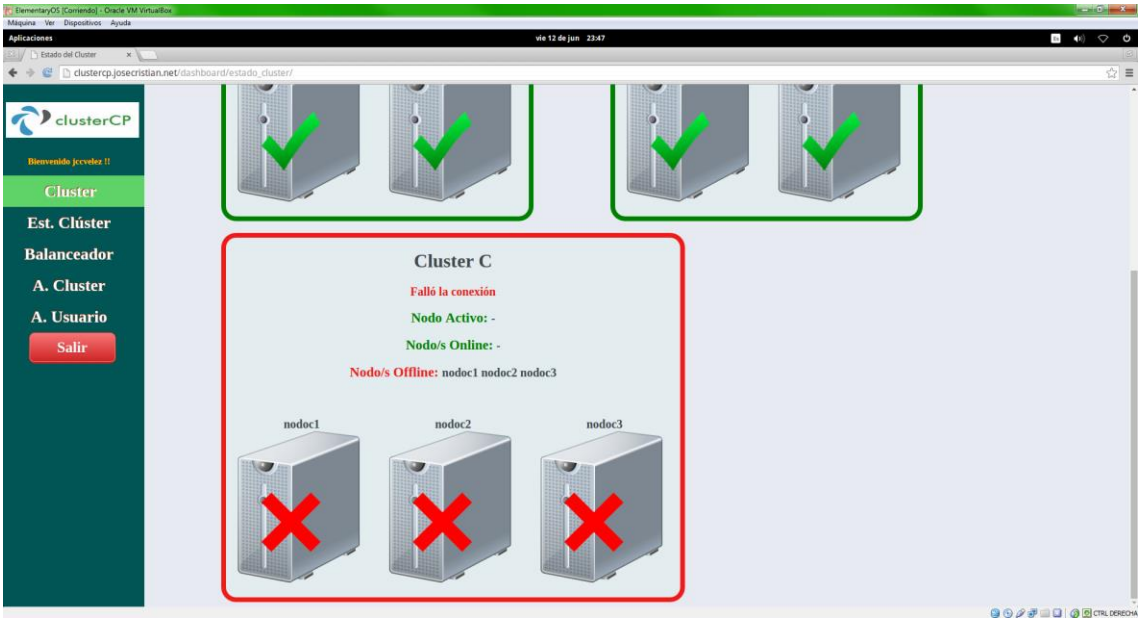


***Imagen 10.6.3. Estado del nuevo clúster.***

Si comprobamos la colección veremos el clúster añadido.

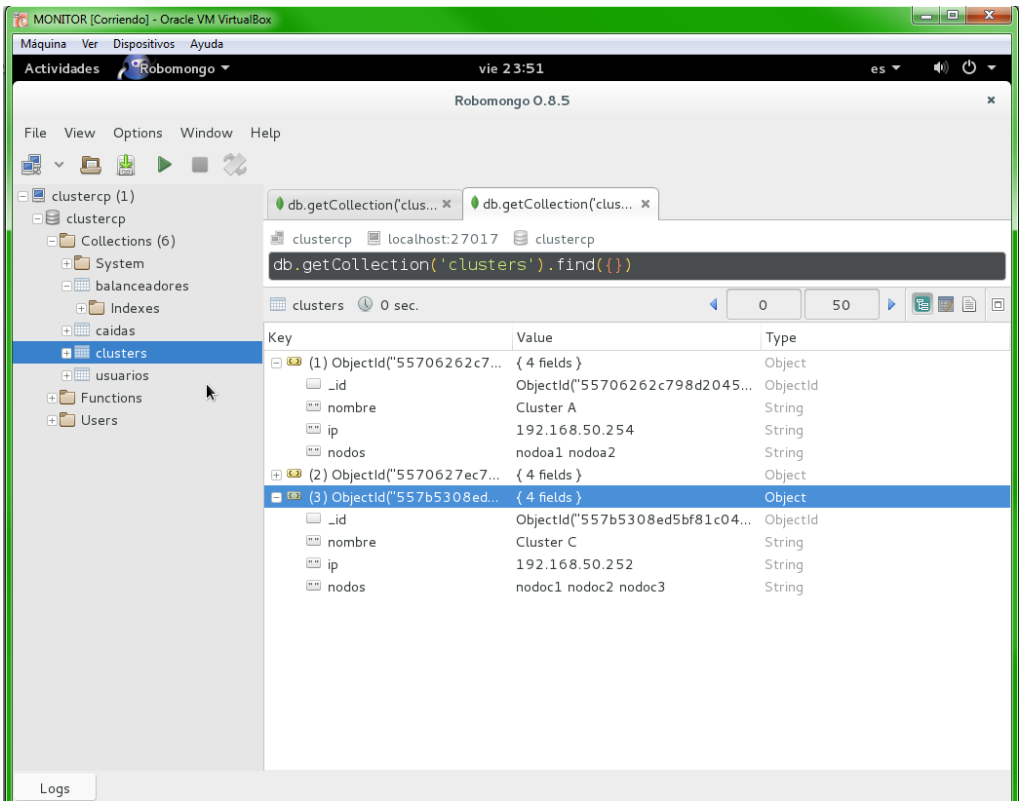


***Imagen 10.6.4. Nuevo cluster añadido a la colección clusters.***

El código del archivo de inicio **/web/dashboard/nuevo_cluster/index.php** es el siguiente.

```
<?php
   // nombre de la pagina para la opcion del menu
   $pagina="nuevo_cluster";
   // Conexion con el controlador
   include("../../../app/controladores/nuevo_cluster_controlador.php");
?>
```

El código del controlador **/app/controladores/nuevo_cluster_controlador.php** que “une” la vista con el modelo es el siguiente.

```
<?php
// incluimos el modelo
include("../../../app/modelos/nuevo_cluster_modelo.php");
// incluimos la vista
include("../../../app/vistas/nuevo_cluster_vista.php");
?>
```

El código del modelo **/app/modelos/nuevo_cluster_modelo.php** que es el encargado de extraer los datos de la BBDD y generar los datos.

```
<?php
   // quitamos en produccion los posibles warning
   error_reporting(0);
   // inclumos el archivo de seguridad
   include("../../../app/includes/seguridad.php");
   // conexion y autenticacion a la BBDD
   include("../../../app/db/conexion.php");
   // comprobamos que todas las variables tienen un valor
   if (isset($_POST['insertar']) and isset($_POST['nombrecluster']) and
isset($_POST['ip']) and isset($_POST['nodos']) )
   {
      // seleccionamos la coleccion clusters
      $coleccion=$db->clusters;
      // recogida del nombre del cluster, ip y nodos
      $nombrecluster=$_POST["nombrecluster"];
      $ip=$_POST["ip"];
      $nodos=$_POST["nodos"];
      // array para insertar lo introducido por el usuario

$arrayinsertar=array("nombre"=>$nombrecluster,"ip"=>$ip,"nodos"=>$no
dos);
      // insertamos los datos
      $insertar=$coleccion->insert($arrayinsertar);
      // si todo esta correcto
         if($insertar){
            // guardamos una variable de ok
```

```
            $estadocreado="si";
            $mensajecreado="Clúster creado correctamente";
        }else{
            // guardamos una variable de error
            $estadocreado="no";
            $mensajecreado="Error creando el clúster";
        }
    }
?>
```

El código de la vista **/app/vistas/nuevo_cluster_vista.php** que muestra el contenido al usuario, es decir, el estilo de los datos.

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8">
        <title>Creacion de Clústers</title>
        <!-- estilo del menu y fondo -->
        <link rel="styleSheet" href="../css/general.css" type="text/css">
        <!-- Estilo del formulario -->
        <link rel="styleSheet" href="../css/form.css" type="text/css">
        <!-- Comprobacion del formulario -->
        <script src="../js/validador_nuevo_cluster.js" type="text/javascript"
></script>
    </head>
    <body>
        <div id="padre">
            <!-- Incluimos el menu -->
            <?php include_once("../includes/sidebar.php") ?>
            <div id="contenido">
                <center>
                    <table>
                        <tbody>
                            <tr>
                                <td style="padding: 20px 0 0 0;">
                                    <form method="post" action="" name="formulario"
onsubmit="return valida()">
                                        <table class="caja_reset">
                                            <center>
                                                <h2>Añadir Nuevo Clúster</h2>
                                            </center>
                                            <tbody>
                                                <tr>
                                                    <td style="padding: 12px 0 0
2px;">Nombre</td>
                                                </tr>
                                                <tr>
                                                    <td><input tabindex="1" type="text"
size="20px" style="width:240px;" name="nombrecluster"
```

```
class="input_reset" id="camponombre" required
onKeyUp="comprobarnombre()"></td>
                                    </tr>
                                    <tr>
                                       <td><div id="statusnombre"></div></td>
                                    </tr>
                                    <tr>
                                       <td style="padding: 12px 0 0 2px;">IP<span
style="padding:0 0 0 14px;"></span></td>
                                    </tr>
                                    <tr>
                                       <td><input tabindex="2" type="text"
size="20px" style="width:240px;" name="ip" class="input_reset"
id="campoip" required onKeyUp="comprobarip()"></td>
                                    </tr>
                                    <tr>
                                       <td><div id="statusip"></div></td>
                                    </tr>
                                    <tr>
                                       <td style="padding: 12px 0 0 2px;">Nodos
(separados con un espacio)<span style="padding:0 0 0
14px;"></span></td>
                                    </tr>
                                    <tr>
                                       <td><input tabindex="2" type="text"
size="20px" style="width:240px;" name="nodos" class="input_reset"
id="camponodos" required onKeyUp="comprobarnodos()"></td>
                                    </tr>
                                    <tr>
                                       <td><div id="statusnodos"></div></td>
                                    </tr>
                                    <tr>
                                       <td height="28px"></td>
                                    </tr>
                                    <tr>
                                       <td style="padding: 0 0 12px 0;"><center>
<input tabindex="3" type="submit" value="Insertar" name="insertar"
class="boton_reset"></center></td>
                                    </tr>
                                 </tbody>
                              </table>
                           </form>
                        </td>
                     </tr>
                     <tr>
                        <td colspan="2">
                           <?php
                              // comprobamos tenga valor
                              if (isset($mensajecreado) and
isset($estadocreado)){
```

```
                                // si se ha creado
                                if ($estadocreado="si"){
                                    echo '<div
class="reset_ok">'.$mensajecreado.'</div>';
                                    // si no se ha creado
                                }else{
                                    echo '<div
class="login_error">'.$mensajecreado.'</div>';
                                }
                            }
                        ?>

                            </td>
                        </tr>
                    </tbody>
                </table>
            </center>
        </div>
        <br style="clear:both"/>
    </div>
  </body>
</html>
```

Para la validación del formulario se ha usado javascript, realizando una comprobación en cada valor introducido.

El código del archivo **/web/dashboard/js/validador_nuevo_cluster.js** que realiza la validación es el siguiente.

```
// funcion global al enviar el formulario, comprobar todos los campos

function valida(){
// nombre cluster
// cogemos el div vacio
estadonombre=document.getElementById("statusnombre");
// extraemos el valor introducido en el campo input
camponombre=document.getElementById("camponombre").value;
// comprobamos el valor introducido
if(!/^[a-zñÑáéíóúÁÉÍÓÚ\s]{3,10}$/i.test(camponombre)){
        // si el valor es diferente ponemos el foco del puntero en el campo
        document.formulario.camponombre.focus();
        // le damos un color de letra
        estadonombre.style.color="#F00";
        // estilo negrita
        estadonombre.style.fontWeight="bold";
        // añadimos un mensaje al HTML
        estadonombre.innerHTML="Debe introducir un nombre correcto";
        // retornamos un error
```

```
        return false;
}else{
        // si todo esta bien le damos color verde
        estadonombre.style.color="rgb(19, 146, 19)";
        // estilo negrita
        estadonombre.style.fontWeight="bold";
        // mensaje
        estadonombre.innerHTML="Correcto";
}
// IP
// cogemos el div vacio
estadoip=document.getElementById("statusip");
// extraemos el valor introducido en el campo input
campoip=document.getElementById("campoip").value;
// comprobamos el valor introducido
if(!/^([1-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])(\.([0-9]|[1-9][0-9]|1[0-9][0-
9]|2[0-4][0-9]|25[0-5])){2}(\.([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-
5]))$/i.test(campoip)){
        // si el valor es diferente ponemos el foco del puntero en el campo
        document.formulario.campoip.focus();
        // le damos un color de letra
        estadoip.style.color="#F00";
        // estilo negrita
        estadoip.style.fontWeight="bold";
        // añadimos un mensaje al HTML
        estadoip.innerHTML="De introducir una IP correcta";
        // retornamos un error
        return false;
}else{
        // si todo esta bien le damos color verde
        estadoip.style.color="rgb(19, 146, 19)";
        // estilo negrita
        estadoip.style.fontWeight="bold";
        // mensaje
        estadoip.innerHTML="Correcto";
}

// nodos
// cogemos el div vacio
estadonodos=document.getElementById("statusnodos");
// extraemos el valor introducido en el campo input
camponodos=document.getElementById("camponodos").value;
// comprobamos el valor introducido
if(!/[A-Za-z\s\t]{3,50}/.test(camponodos)){
        // si el valor es diferente ponemos el foco del puntero en el campo
        document.formulario.camponodos.focus();
        // le damos un color de letra
        estadonodos.style.color="#F00";
        // estilo negrita
        estadonodos.style.fontWeight="bold";
```

```
        // añadimos un mensaje al HTML
        estadonodos.innerHTML="Debe introducir un nombre correcto";
        // retornamos un error
        return false;
}else{
        // si todo esta bien le damos color verde
        estadonodos.style.color="rgb(19, 146, 19)";
        // estilo negrita
        estadonodos.style.fontWeight="bold";
        // mensaje
        estadonodos.innerHTML="Correcto";
}

}

//// campos sueltos - onkeyup

// nombre cluster
function comprobarnombre(){
// cogemos el div vacio
estadonombre=document.getElementById("statusnombre");
// extraemos el valor introducido en el campo input
camponombre=document.getElementById("camponombre").value;
// comprobamos el valor introducido
if(!/^[a-zñÑáéíóúÁÉÍÓÚ\s]{3,10}$/i.test(camponombre)){
        // si el valor es diferente ponemos el foco del puntero en el campo
        document.formulario.camponombre.focus();
        // le damos un color de letra
        estadonombre.style.color="#F00";
        // estilo negrita
        estadonombre.style.fontWeight="bold";
        // añadimos un mensaje al HTML
        estadonombre.innerHTML="Debe introducir un nombre correcto";
        // retornamos un error
        return false;
}else{
        // si todo esta bien le damos color verde
        estadonombre.style.color="rgb(19, 146, 19)";
        // estilo negrita
        estadonombre.style.fontWeight="bold";
        // mensaje
        estadonombre.innerHTML="Correcto";
}
}
// IP
function comprobarip(){
// cogemos el div vacio
estadoip=document.getElementById("statusip");
// extraemos el valor introducido en el campo input
campoip=document.getElementById("campoip").value;
```

```
// comprobamos el valor introducido
if(!/^([1-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])(\.([0-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])){2}(\.([0-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5]))$/i.test(campoip)){
        // si el valor es diferente ponemos el foco del puntero en el campo
        document.formulario.campoip.focus();
        // le damos un color de letra
        estadoip.style.color="#F00";
        // estilo negrita
        estadoip.style.fontWeight="bold";
        // añadimos un mensaje al HTML
        estadoip.innerHTML="De introducir una IP correcta";
        // retornamos un error
        return false;
}else{
        // si todo esta bien le damos color verde
        estadoip.style.color="rgb(19, 146, 19)";
        // estilo negrita
        estadoip.style.fontWeight="bold";
        // mensaje
        estadoip.innerHTML="Correcto";
}
}
// nodos
function comprobarnodos(){
// cogemos el div vacio
estadonodos=document.getElementById("statusnodos");
// extraemos el valor introducido en el campo input
camponodos=document.getElementById("camponodos").value;
// comprobamos el valor introducido
if(!/[A-Za-z\s\t]{3,50}/.test(camponodos)){
        // si el valor es diferente ponemos el foco del puntero en el campo
        document.formulario.camponodos.focus();
        // le damos un color de letra
        estadonodos.style.color="#F00";
        // estilo negrita
        estadonodos.style.fontWeight="bold";
        // añadimos un mensaje al HTML
        estadonodos.innerHTML="Debe introducir un nombre correcto";
        // retornamos un error
        return false;
}else{
        // si todo esta bien le damos color verde
        estadonodos.style.color="rgb(19, 146, 19)";
        // estilo negrita
        estadonodos.style.fontWeight="bold";
        // mensaje
        estadonodos.innerHTML="Correcto";
}
}
```

## 10.7 Añadir Usuario

Para crear usuarios que accedan al panel de monitorización haremos clic en el menú **A. Usuario** y accederemos a un formulario en el que añadiremos el usuario y el password.

Los usuarios añadidos serán administradores, por lo que podrán añadir más usuarios y tendrán control sobre todo el sistema ya que este sistema de monitorización está orientado a administradores de sistemas y no a usuarios finales.

Añadiremos un usuario **josecristian** con contraseña **claveinves**

La contraseña se guardará en formato MD5 en la colección **usuarios** a través de un **insert**

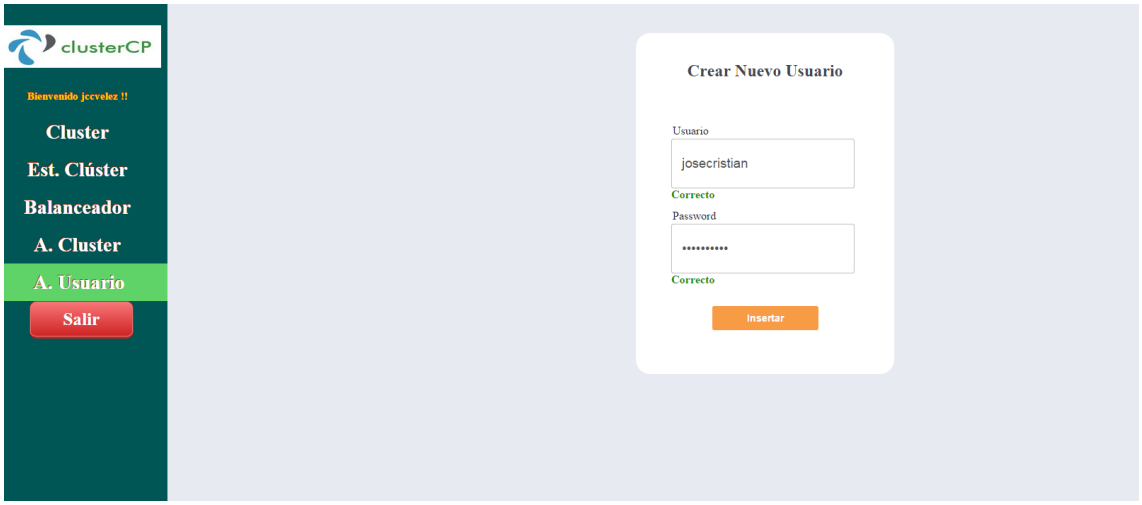


***Imagen 10.7.1. Creación de nuevo usuario.***

Si el usuario se añade correctamente veremos un mensaje.

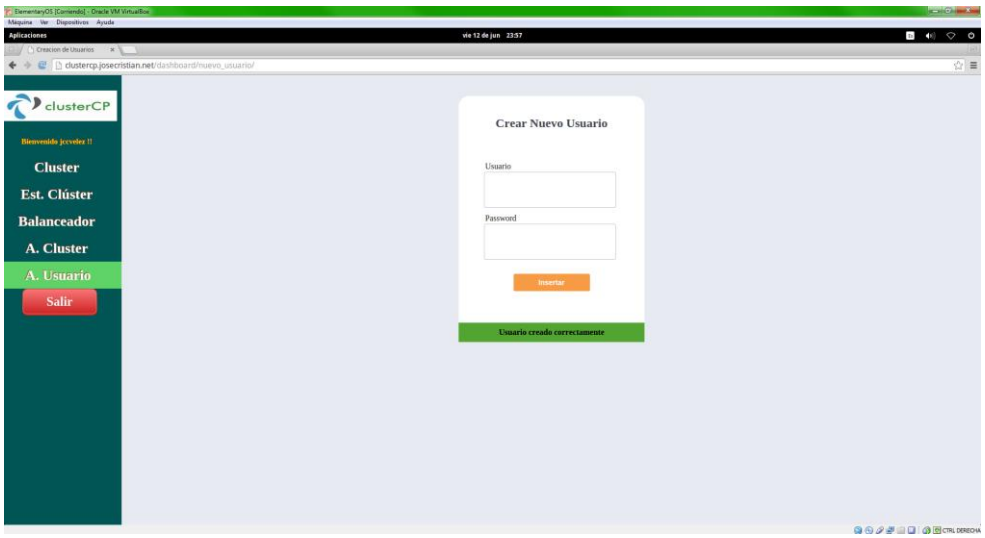


***Imagen 10.7.2. Usuario creado correctamente.***

El código del archivo de inicio **/web/dashboard/nuevo_usuario/index.php** es el siguiente.

```
<?php
   // nombre de la pagina para el menu
   $pagina='nuevousuario';
   // Conexion con el controlador
   include("../../../app/controladores/nuevousuario_controlador.php")
?>
```

El código del controlador **/app/controladores/nuevousuario_controlador.php** que “une” la vista con el modelo es el siguiente.

```
<?php
        // incluimos el modelo
        include '../../../app/modelos/nuevousuario_modelo.php';
        // incluimos la vista
        include '../../../app/vistas/nuevousuario_vista.php';
?>
```

El código del modelo /**app/modelos/nuevousuario_modelo.php** que es el encargado de extraer los datos de la BBDD y generar los datos.

```
<?php
   // quitamos en produccion los posibles warning
   error_reporting(0);
   // archivo de seguridad
   include("../../../app/includes/seguridad.php");
   // conexion y autenticacion a la BBDD
   include('../../../app/db/conexion.php');
   // comprobamos que todas las variables tienen un valor
   if (isset($_POST['insertar']) and isset($_POST['usuario']) and
isset($_POST['password']) )
   {
      // seleccionamos la coleccion usuarios
      $coleccion=$db->usuarios;
      // recogida el usuario y password por POST
      $usuario=$_POST["usuario"];
      // convertimos el password a MD5
      $password=md5($_POST["password"]);
      // array para insertar lo introducido por el usuario
      $arrayinsertar=array("usuario"=>$usuario,"password"=>$password);
      // insertamos los datos
      $insertar=$coleccion->insert($arrayinsertar);
      // si todo esta correcto
         if($insertar){
            // guardamos una variable de ok
            $estadocreado="si";
            $mensajecreado="Usuario creado correctamente";
```

```
        }else{
            // guardamos una variable de error
            $estadocreado="no";
            $mensajecreado="Error creando el usuario";
        }
    }
?>
```

El código de la vista **/app/vistas/nuevousuario_vista.php** que muestra el contenido al usuario, es decir, el estilo de los datos.

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8">
        <title>Creacion de Usuarios</title>
        <!-- estilo del menu y fondo-->
        <link rel="styleSheet" href="../css/general.css" type="text/css">
        <!-- Estilo del formulario-->
        <link rel="styleSheet" href="../css/form.css" type="text/css">
        <!-- Comprobacion del formulario -->
        <script src="../js/validador_nuevo_usuario.js" type="text/javascript"
></script>
    </head>
    <body>
        <div id="padre">
            <!-- Incluimos el menu -->
            <?php include("../includes/sidebar.php") ?>
            <div id="contenido">
            <center>
                <table>
                    <tbody>
                        <tr>
                            <td style="padding: 20px 0 0 0;">
                                <form method="post" action="" name="formulario"
onsubmit="return valida()">
                                    <table class="caja_reset">
                                        <center>
                                            <h2>Crear Nuevo Usuario</h2>
                                        </center>
                                        <tbody>
                                            <tr>
                                                <td style="padding: 12px 0 0
2px;">Usuario</td>
                                            </tr>
                                            <tr>
                                                <td><input tabindex="1" type="text"
size="20px" style="width:240px;" name="usuario" class="input_reset"
id="campousuario" required onKeyUp="comprobarusuario()"></td>
                                            </tr>
```

```
                                <tr>
                                    <td><div id="statususuario"></div></td>
                                </tr>
                                <tr>
                                    <td style="padding: 12px 0 0
2px;">Password<span style="padding:0 0 0 14px;"></span></td>
                                </tr>
                                <tr>
                                    <td><input tabindex="2" type="password"
size="20px" style="width:240px;" name="password" class="input_reset"
id="newpass" required onKeyUp="comprobarpassword()"></td>
                                </tr>
                                <tr>
                                    <td><div id="statuscontrasenia"></div></td>
                                </tr>
                                <tr>
                                    <td height="28px"></td>
                                </tr>
                                <tr>
                                    <td style="padding: 0 0 12px 0;"><center>
<input tabindex="3" type="submit" value="Insertar" name="insertar"
class="boton_reset"></center></td>
                                 </tr>
                            </tbody>
                        </table>
                    </form>
                </td>
            </tr>
            <tr>
                <td colspan="2">
                    <?php
                      // comprobamos tenga valor
                      if (isset($mensajecreado) and isset($estadocreado)){
                         // si se ha creado
                         if ($estadocreado="si"){
                            echo '<div
class="reset_ok">'.$mensajecreado.'</div>';
                                // si no se ha creado
                            }else{
                               echo '<div
class="login_error">'.$mensajecreado.'</div>';
                            }
                        }
                    ?>

                </td>
            </tr>
        </tbody>
    </table>
```

```
        </center>
        </div>
        <br style="clear:both"/>
      </div>
   </body>
</html>
```

Para la validación del formulario se ha usado javascript, realizando una comprobación en cada valor introducido.

El código del archivo **/web/dashboard/js/validador_nuevo_usuario.js** que realiza la validación es el siguiente.

```
// funcion global al enviar el formulario, comprobar todos los campos

function valida(){
        // usuario
        // cogemos el div vacio
        estadousuario=document.getElementById("statususuario");
        // extraemos el valor introducido en el campo input
        campousuario=document.getElementById("campousuario").value;
        // comprobamos el valor introducido
        if(!/^[a-zñÑáéíóúÁÉÍÓÚ\s]{3,15}$/i.test(campousuario)){
                // si el valor es diferente ponemos el foco del puntero en el
campo
                document.formulario.usuario.focus();
                // le damos un color de letra
                estadousuario.style.color="#F00";
                // estilo negrita
                estadousuario.style.fontWeight="bold";
                // añadimos un mensaje al HTML
                estadousuario.innerHTML="Debe introducir un usuario
correcto";
                // retornamos un error
                return false;
        }else{
                // si todo esta bien le damos color verde
                estadousuario.style.color="rgb(19, 146, 19)";
                // estilo negrita
                estadousuario.style.fontWeight="bold";
                // mensaje
                estadousuario.innerHTML="Correcto";
        }

        //password
        // cogemos el div vacio
        estadopass=document.getElementById("statuscontrasenia");
```

```
        // extraemos el valor introducido en el campo input
        campopass=document.getElementById("newpass").value;
        // comprobamos el valor introducido
        if(!/^[a-zA-Z0-9\s]{4,10}$/i.test(campopass)){
                // si el valor es diferente ponemos el foco del puntero en el
campo
                document.formulario.newpass.focus();
                // le damos un color de letra
                estadopass.style.color="#F00";
                // estilo negrita
                estadopass.style.fontWeight="bold";
                // añadimos un mensaje al HTML
                estadopass.innerHTML="4 a 10 caracteres y solo numeros y
letras";
                // retornamos un error
                return false;
        }else{
                // si todo esta bien le damos color verde
                estadopass.style.color="rgb(19, 146, 19)";
                // estilo negrita
                estadopass.style.fontWeight="bold";
                // mensaje
                estadopass.innerHTML="Correcto";
        }

}

//// campos sueltos - onkeyup

// usuario
function comprobarusuario(){
        // cogemos el div vacio
        estadousuario=document.getElementById("statususuario");
        // extraemos el valor introducido en el campo input
        campousuario=document.getElementById("campousuario").value;
        // comprobamos el valor introducido
        if(!/^[a-zñÑáéíóúÁÉÍÓÚ\s]{3,15}$/i.test(campousuario)){
                // si el valor es diferente ponemos el foco del puntero en el
campo
                document.formulario.usuario.focus();
                // le damos un color de letra
                estadousuario.style.color="#F00";
                // estilo negrita
                estadousuario.style.fontWeight="bold";
                // añadimos un mensaje al HTML
                estadousuario.innerHTML="Debe introducir un usuario
correcto";
                // retornamos un error
                return false;
        }else{
```

```
                // si todo esta bien le damos color verde
                estadousuario.style.color="rgb(19, 146, 19)";
                // estilo negrita
                estadousuario.style.fontWeight="bold";
                // mensaje
                estadousuario.innerHTML="Correcto";
        }
}
// passuno

function comprobarpassword(){
        // cogemos el div vacio
        estadopass=document.getElementById("statuscontrasenia");
        // extraemos el valor introducido en el campo input
        campopass=document.getElementById("newpass").value;
        // comprobamos el valor introducido
        if(!/^[a-zA-Z0-9\s]{4,10}$/i.test(campopass)){
                // si el valor es diferente ponemos el foco del puntero en el
campo
                document.formulario.newpass.focus();
                // le damos un color de letra
                estadopass.style.color="#F00";
                // estilo negrita
                estadopass.style.fontWeight="bold";
                // añadimos un mensaje al HTML
                estadopass.innerHTML="4 a 10 caracteres y solo numeros y
letras";
                // retornamos un error
                return false;
        }else{
                // si todo esta bien le damos color verde
                estadopass.style.color="rgb(19, 146, 19)";
                // estilo negrita
                estadopass.style.fontWeight="bold";
                // mensaje
                estadopass.innerHTML="Correcto";
        }
}
```

## 10.8 Monitorización de HAproxy

El panel de monitorización de HAproxy permite visualizar las peticiones a cada **cluster** y al penel **clustercp**, para acceder deberemos pulsar en el menú la sección **balanceador** que redirigirá a la página [http://clustercp.josecristian.net/haproxy?stats](http://clustercp.josecristian.net/haproxy?stats)

Al acceder a la página nos pedirá usuario y contraseña que será la que configuramos en el archivo **/etc/haproxy/haparoxy.conf** a través de las directivas:

```
stats enable
stats auth jccvelez:inves
```

Una vez dentro veremos las estadísticas.

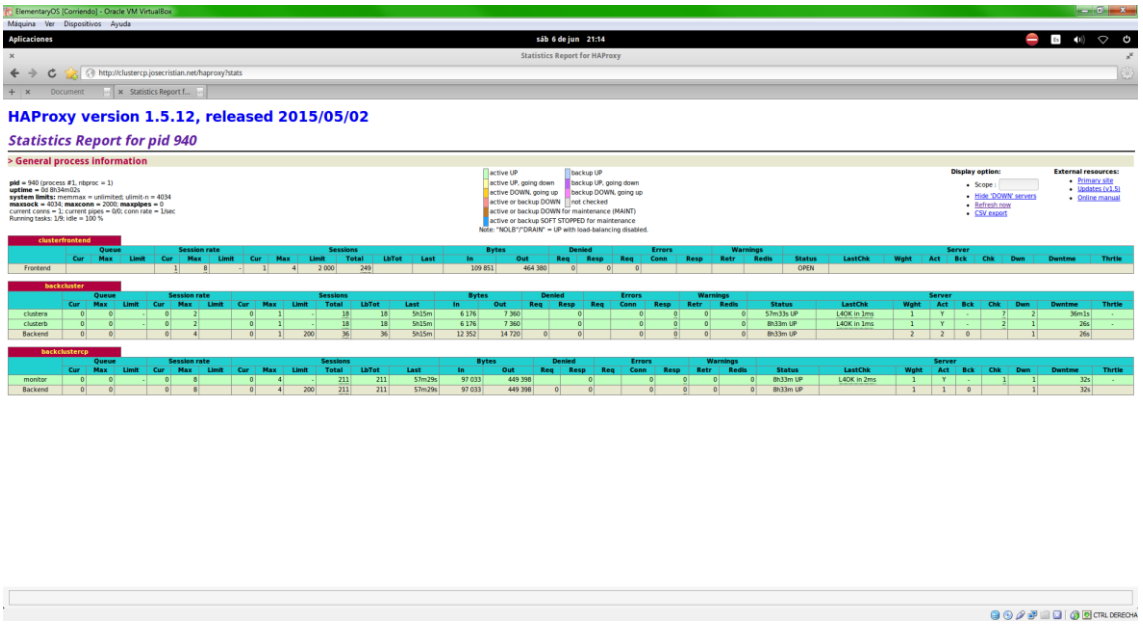


***Imagen 10.8.1. Panel de estadísticas del balanceador de carga.***

Veremos las sesiones de cada nodo del clúster, el color verde indica que el clúster está operativo o uno de sus nodos, si esta rojo indica que han fallado todos los nodos del clúster, también podremos ver el ancho de banda y el uptime.

## 10.9 Notificación ante fallos en el clúster.

Tendremos un script PHP que se ejecutara cada cierto tiempo a través de una tarea programada usando **cron**, dicho script se ejecutara con **CURL.**

Antes de comenzar con el script deberemos instalar el paquete **SSMTP** para que podamos enviar emails a través de la función **mail** de PHP.

```
$sudo apt-get install ssmpt
```

Editamos el archivo **/etc/ssmtp/ssmtp.conf** y añadimos las credenciales del correo desde que se enviaran las notificaciones.

```
# Servidor SMTP al que reenviaremos los correos.
mailhub=smtp.gmail.com:587
# Datos de autenticación de la cuenta de correo.
AuthUser=notificaciones.clustercp@gmail.com
AuthPass=Claveinves12
# Usar SSL/TLS
UseTLS=Yes
UseSTARTTLS=Yes
```

Ahora instalamos CURL y la extensión para PHP.

```
$sudo apt-get install curl php5-curl
```

El script alojado en /var/www/html/scripts denominado **notificaciones_cluster.php**, comprueba en primer lugar si realiza ping a cada cluster (las direcciones IP son extraídas de la colección **clusters**), si no enviamos un correo con el nombre del clúster extraído de la colección **clusters** asociado a la dirección IP (usando la misma posición de cada array, arrayip, arraynombre) esta notificación por correo electrónico está en la función **noti_ping** usando la función **mail** a la dirección de correo indicada en la variable **$correonotificacion** y guardamos un log con el error en el archivo **/var/log/cluster/error_cluster.log** usando la función **notificaciones_log**.

Si se realiza ping comprobamos la conexión SSH, si no realizamos conexión SSH llamamos a la función **notificacioncaida** que realiza el envío de un correo con los nodos extraídos de la colección y el nombre del clúster asociados a la IP que se está comprobando, la fecha y la hora, también se realiza una inserción en la colección **caídas** de cada nodo fallido, la fecha y la hora.

Si hay un error en la autenticación SSH guardamos el error con el nombre del clúster en el log con la fecha y hora.

Si no hay error en la autenticación comprobamos los nodos caídos a través del comando **pcs status nodes** y extraemos los nodos offline de la línea 4 y llamamos a la función **notificacioncaida** a través de la cual se enviará el correo con el nodo/s fallido/s, el nombre del clúster, la fecha y la hora y se registrará en el log.

Programaremos la ejecución del script PHP con **crontab** cada minuto usando **CURL.**

```
#crontab -e
```

Añadimos:

```
* * * * * /usr/bin/curl –u josecristian:claveinves http://localhost/scripts/notificacion_cluster.php
```

**NOTA: Con el parámetro –u entramos al directorio protegido con usuario y contraseña para evitar que cualquier usuario ejecute el archivo.**

Posteriormente protegeremos el directorio a través de apache, primero crearemos el usuario usando el comando **htpasswd** y crearemos un archivo **/etc/apache2/usuarios**

**htpasswd –c /etc/apache/usuarios josecristian**

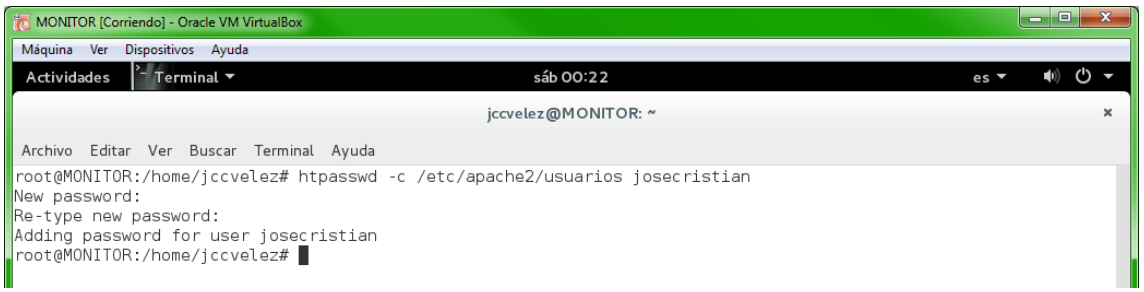


***Imagen 10.9.1. Usuario de acceso al directorio del script de notificacion.***

Una vez creado el usuario añadiremos la protección al directorio editando el archivo de configuración de la página de apache, además restringiremos el acceso únicamente desde **localhost** y añadiremos una sección **Directory.**

```
        #Include conf-available/serve-cgi-bin.conf
        <Directory /var/www/html/web/scripts>
        AuthType basic
        AuthName "Acceso privado"
        AuthUserFile "/etc/apache2/usuarios"
        Require user josecristian
        Order allow,deny
        Allow from localhost
        </Directory>
</VirtualHost>
```

***Imagen 10.9.2. Directorio protegido por contraseña.***

Si falla un nodo recibiremos un mensaje como el siguiente:

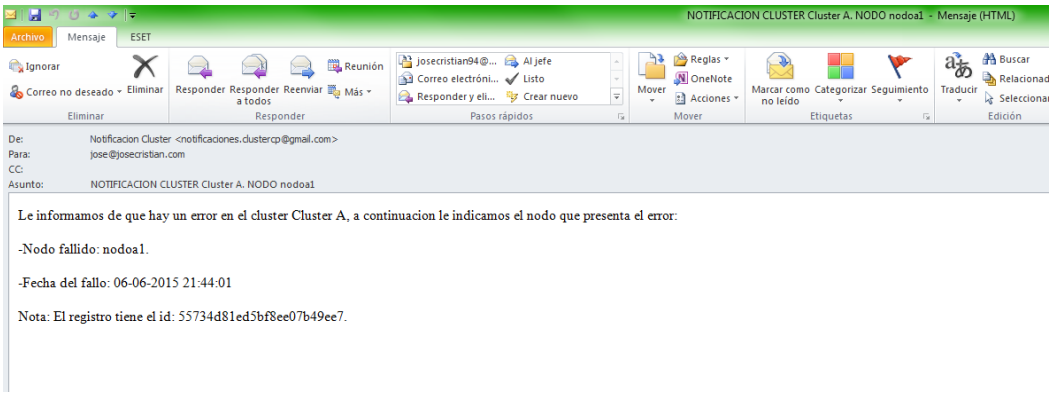


***Imagen 10.9.3. Notificación del fallo por correo electronico.***

Revisaremos el log.

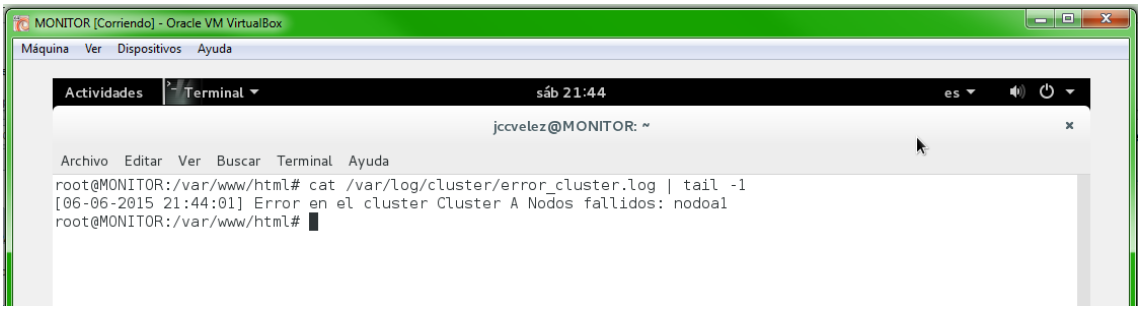


***Imagen 10.9.4. Fallo registrado en el log.***

También revisaremos las estadísticas y el panel del clúster para comprobarlo.

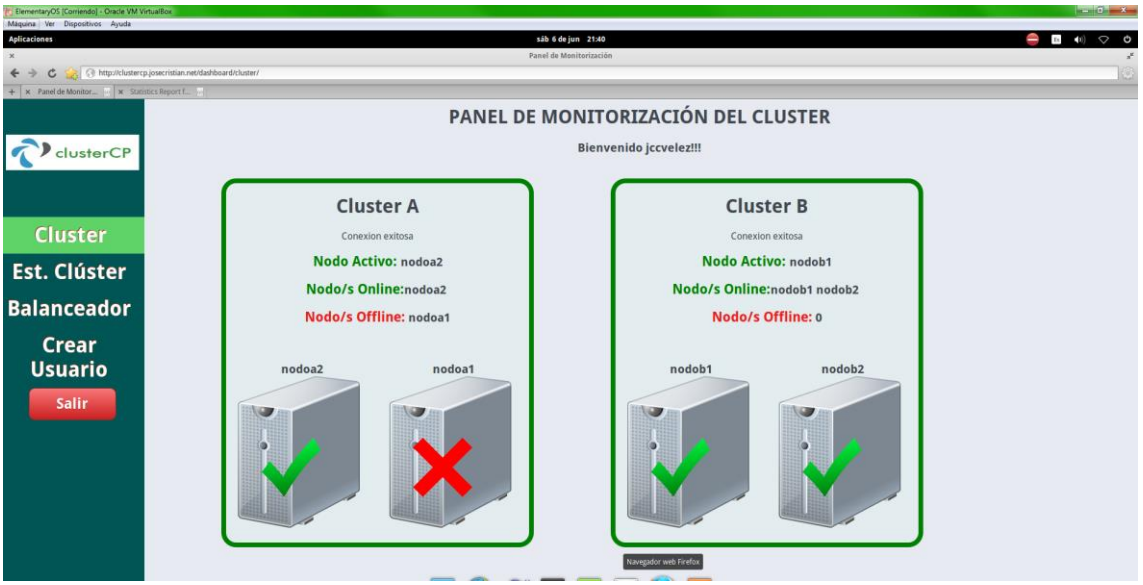


***Imagen 10.9.5. Error del nodo.***

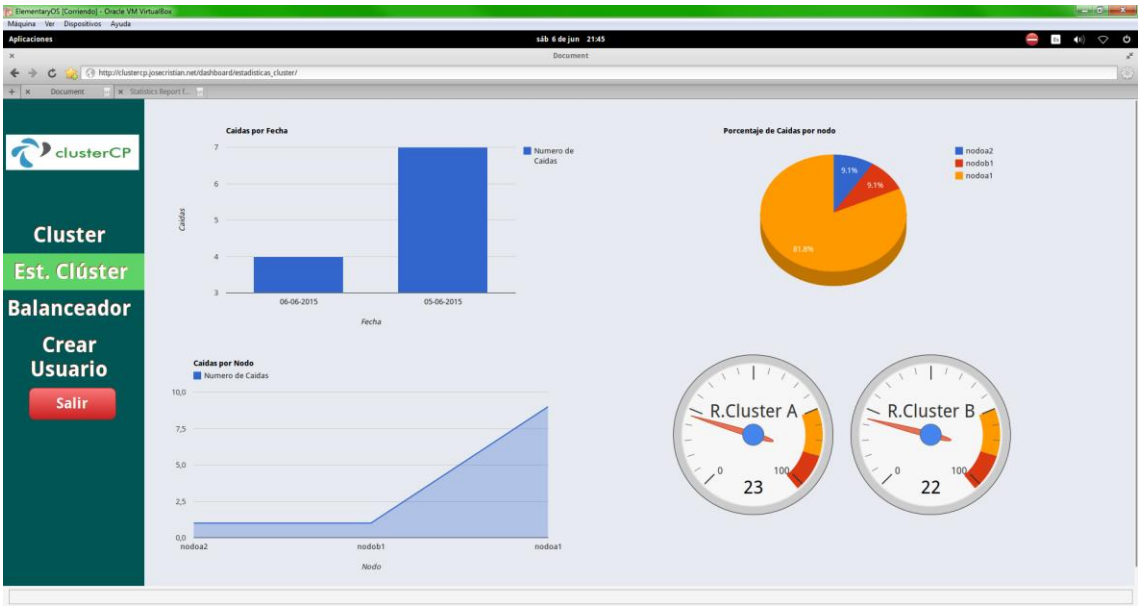


*Imagen 10.9.6. Estadisticas de nodos caidos y RAM usada en cada clúster.*

El archivo **/var/www/html/scripts/notificación_cluster.php** contiene el siguiente código.

```
<?php
   // quitamos los warning
   error_reporting(0);
   // Conexion BBDD
   $conexion = new Mongo();
   $db = $conexion->clustercp;
   $db->authenticate('jose', 'inves');
   // accedemos a la bbdd caidas
   $coleccioncaidas=$db->caidas;
   // accedemos a la bbdd clusters
   $coleccionclusters=$db->clusters;
   // definimos los arrays
   $arrayip=array();
   $arraynombre=array();
   $arraynodos=array();
   // realizamos una consulta de todos los documentos
   $clusters=$coleccionclusters->find();
   // por cada documento guardamos cada valor
   // en la posicion adeacuada de cada array
   foreach ($clusters as $datosclusters) {
      foreach ($datosclusters as $columna=>$datoscluster) {
         if ($columna=="ip") array_push($arrayip, $datoscluster);
         if ($columna=="nombre") array_push($arraynombre,
$datoscluster);
         if ($columna=="nodos") array_push($arraynodos, $datoscluster);
      }
   }
```

```
   // correo de notificacion
   $correonotificacion="jose@josecristian.com";

    //CLUSTER A

   // notificacion email ping

   function noti_ping($idcluster,$correo){

      $fecha_hora_actual=date("d-m-Y H:i:s");

      $para = $correo;
      // Asunto
      $titulo = 'NOTIFICACION CLUSTER '.$idcluster;
      // Mensaje
      $mensaje = '
      <html>
      <head>
      <title>Error en el cluster '.$idcluster.'</title>
      </head>
      <body>
      <p>No se ha podido establecer la conexion con el cluster
'.$idcluster.'.</p>
      <p>-Fecha del fallo: '.$fecha_hora_actual.'</p>
      </body>
      </html>
      ';
      // Cabecera que especifica que es un HMTL
      $cabeceras  = 'MIME-Version: 1.0' . "\r\n";
      $cabeceras .= 'Content-type: text/html; charset=iso-8859-1' . "\r\n";
      // Remitente
      $cabeceras .= 'From: Notificacion Cluster
<noti_cluster@josecristian.com>' . "\r\n";

      //Enviamos el email y comprobamos si se envia o no
      if (mail($para, $titulo, $mensaje, $cabeceras)){
         $mensajelog="Email enviado correctamente";
         notificaciones_log("ok",$mensajelog,date("d-m-Y H:i:s"));
      }else{
         $mensajelog="Error al enviar el Email";
         notificaciones_log("error",$mensajelog,date("d-m-Y H:i:s"));
      }
   }

   // notificacion cluster

   function notificacioncaida($nodos,$idcluster,$coleccion,$correo){
```

```
        // Comprobamos el ping, si hacemos ping seguimos, si no
enviamos un correo con el error del ping.

        // creamos un array de los nodos de forma individual separando
los nodos por espacios para enviar insertar una fila por nodo
        $nodos=str_replace(" ",",",$nodos);
        $array = explode(",",$nodos);

        // Ahora contamos la longitud del array y generamos un bucle con
la insercion de mysql por cada posicion del array
        $fecha_actual=date("d-m-Y");
        $hora_actual=date("H:i:s");

        foreach ($array as $nodoindividual) {

$queryarray=array("nodo"=>$nodoindividual,"fecha"=>$fecha_actual,"hor
a"=>$hora_actual);
            $insercion=$coleccion->insert($queryarray);

            // guardamos registro en el log log_cluster.log del nodo caido y
la fecha y hora


            if($insercion){
                $ultimoregistro = $coleccion->find()->sort(array("_id" => -1))-
>limit(1);
                foreach ($ultimoregistro as $idregistro) {
                    $idregistro=$idregistro["_id"];
                }
                // enviamos el email con el nodo fallido

                $para = $correo;
                // Asunto
                $titulo = 'NOTIFICACION CLUSTER '.$idcluster.'. NODO
'.$nodoindividual;
                // Cuerpo o mensaje
                $mensaje = '
                <html>
                <head>
                <title>Error en el cluster '.$idcluster.'</title>
                </head>
                <body>
                <p>Le informamos de que hay un error en el cluster
'.$idcluster.', a continuacion le indicamos el nodo que presenta el
error:</p>
                <p>-Nodo fallido: '.$nodoindividual.'.</p>
                <p>-Fecha del fallo: '.$fecha_actual." ".$hora_actual.'</p>
                <p>Nota: El registro tiene el id: '.$idregistro.'.</p>
                </body>
                </html>
```

```
                ';

                // Cabecera que especifica que es un HMTL
                $cabeceras  = 'MIME-Version: 1.0' . "\r\n";
                $cabeceras .= 'Content-type: text/html; charset=iso-8859-1' .
"\r\n";
                // Cabeceras adicionales
                $cabeceras .= 'From: Notificacion Cluster
<noti_cluster@josecristian.com>' . "\r\n";
                if (mail($para, $titulo, $mensaje, $cabeceras)){
                   // guardamos registro en el log ok_mail.log
                }else{
                   // guardamos registro en el log error_mail.log
                }

                // si se envia el correo registramos el envio del correo, si no
un error actualizando el ultimo campo extraido con el id $datos.


            }else{
                // guardamos registro en el log log_cluster.log de error al
realizar el insert
            }
        }
}

// pasamos las variables tipo de log si es de error o de ok

   function notificaciones_log($tipo,$objeto,$fechahora){
      // si la notificacion es de ok guardamos lo obtenido
      // a traves de los parametros de la funcion y añadimos el mensaje
      // en el archivo ok_cluster.log
      if($tipo=="ok"){
         $file = fopen("/var/log/cluster/ok_cluster.log", "a");
         $mensaje="[".$fechahora."] ".$objeto;
         fwrite($file, $mensaje . PHP_EOL);
         fclose($file);
      }

      if($tipo=="error"){
         // si la notificacion es de error guardamos lo obtenido
         // a traves de los parametros de la funcion y añadimos el mensaje
         // en el archivo error_cluster.log
         $file = fopen("/var/log/cluster/error_cluster.log", "a");
         $mensaje="[".$fechahora."] ".$objeto;
         fwrite($file, $mensaje . PHP_EOL);
         fclose($file);
      }
```

```
   }

            // obtenemos la ip de cada array y la posicion de la columna
            // para acceder a la misma posicion a los demas arrays de
            // nombre de cluster y nombre de nodo
            foreach ($arrayip as $columna => $ipcluster) {
            //realizamos el ping
            $ping=exec("ping -c 1 -w 1 ".$ipcluster, $input, $resultado);
            // creamos un array con los nodos obtenidos de MongoDB
            $arraynodosextraido=explode(" ", $arraynodos[$columna]);
               if ($resultado==0) {
                  if(!($con = ssh2_connect($ipcluster, 22))){
                     // Notificamos la caida si no se realiza conexion a
traves de SSH
                     $mensajelog="Error de conexion SSH en el cluster
".$arraynombre[$columna]." Nodo fallido: ".$arraynodos[$columna];
                     // guardamos en el log a traves de la funcion
                     notificaciones_log("error",$mensajelog,date("d-m-Y
H:i:s"));
                     // guardamos en la coleccion a traves de la funcion

notificacioncaida($arraynodos[$columna],$arraynombre[$columna],$cole
ccioncaidas,$correonotificacion);



                  } else {
                     // si no nos autenticamos
                     if(!ssh2_auth_password($con, "root", "inves")) {
                        $mensajelog="Error de autenticacion SSH en el
cluster ".$arraynombre[$columna];
                        // guardamos mensaje en el log a traves de la funcion
y le pasamos el tipo
                        // como error
                        notificaciones_log("error",$mensajelog,date("d-m-Y
H:i:s"));
                     } else {
                           // si nos conectamos a traves de SSH
                           if (!($datos = ssh2_exec($con, "pcs status nodes |
head -4 | tail -1"))) {
                              $mensajelog="Fallo de conexion SSH
obteniendo los datos de los nodos en el cluster
".$arraynombre[$columna];
                              // si hay un error en la ejecucion del comando
                              // guardamos en el log el error a traves de la
funcion
                              notificaciones_log("error",$mensajelog,date("d-
m-Y H:i:s"));
                           } else {
```

```
                                stream_set_blocking($datos, true);
                                $nodosoffline = "";
                                   while ($buf = fread($datos,4096)) {
                                      $nodosoffline .= $buf;
                                   }

                                // filtrado de resultados
                                   // quitamos la palabra Offline:
                                   $nodosoffline=str_replace("Offline:
","",$nodosoffline);
                                   // quitamos los espacios
                                   $nodosoffline=trim($nodosoffline);
                                   // si hay nodos offline
                                   if ($nodosoffline){
                                      // eliminamos los espacios
                                      $nodosoffline=trim($nodosoffline);
                                      // log
                                      $mensajelog="Error en el cluster
".$arraynombre[$columna]." Nodos fallidos: ".$nodosoffline;
                                      // guardamos los mensajes en el log a
traves de la funcion

notificaciones_log("error",$mensajelog,date("d-m-Y H:i:s"));
                                      // enviamos el email de los nodos offline
                                      // a traves de la funcion

notificacioncaida($nodosoffline,$arraynombre[$columna],$coleccioncaida
s,$correonotificacion);
                                   }
                                  // cerramos la conexion
                                   fclose($datos);
                             }
                          }
                       }
                    }else{
                    // si no realizamos
                       $mensajelog="Error haciendo ping al cluster
".$arraynombre[$columna];
                       // guardamos el mensaje en el log con la fecha a traves de
la funcion
                       notificaciones_log("error",$mensajelog,date("d-m-Y
H:i:s"));
                       // enviamos un email a traves de la funcion con el nombre
                       // del array y la fecha
                       noti_ping($arraynombre[$columna],$correonotificacion);

                    }
}

?>
```

# 11. Funcionamiento y pruebas

Comenzaremos con el cluster funcionando correctamente, todos los nodos, balanceadores,etc. funcionan correctamente.

Accederemos a la pagina [http://clustercp.josecristian.net](http://clustercp.josecristian.net) y nos loguearemos.

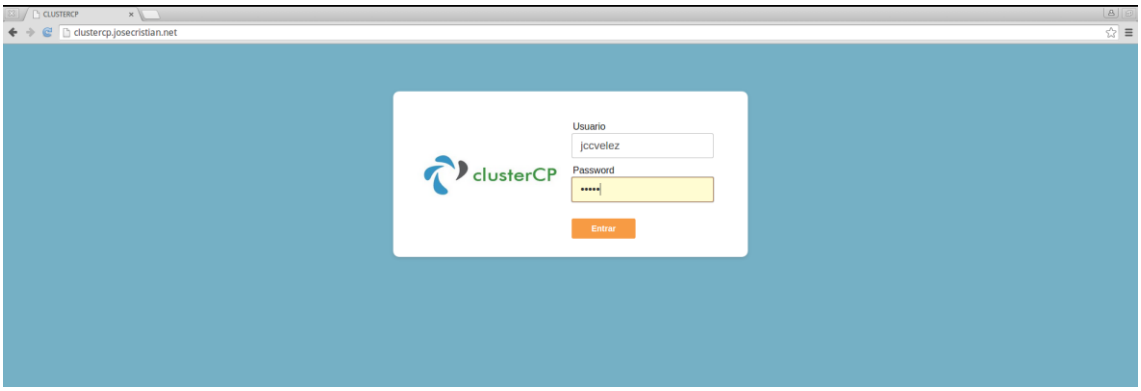


***Imagen 11.1. Login de acceso al panel.***

Una vez logueados accederemos a la pagina inicial de monitorización del clúster donde veremos que todo esta correctamente.

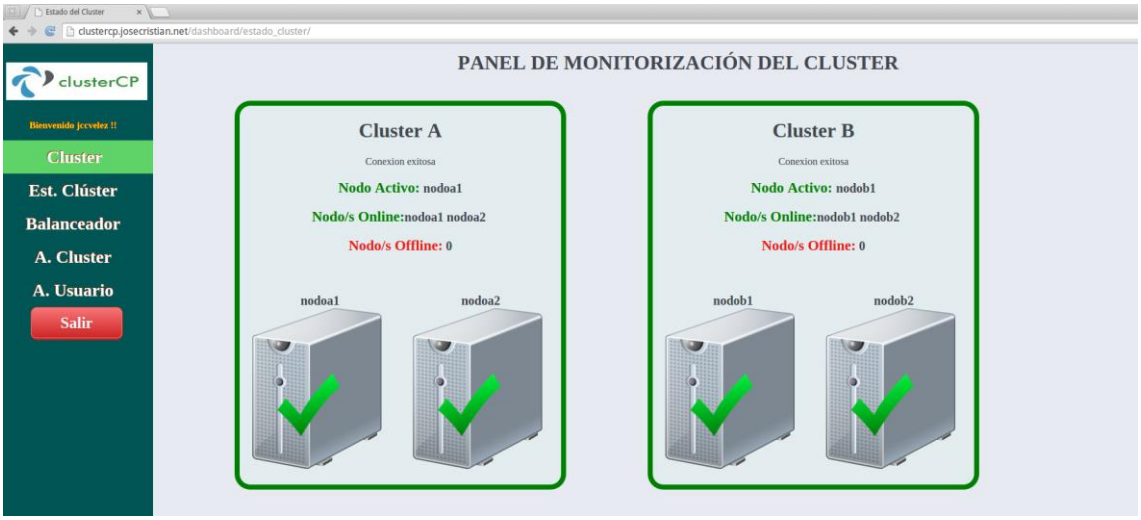


***Imagen 11.2. Estado de los nodos.***

Nota: veremos el nodo activo de cada cluster.

Ahora accederemos a la página web que ofrecen los clusters con el contenido replicado usando **GlusterFS.**

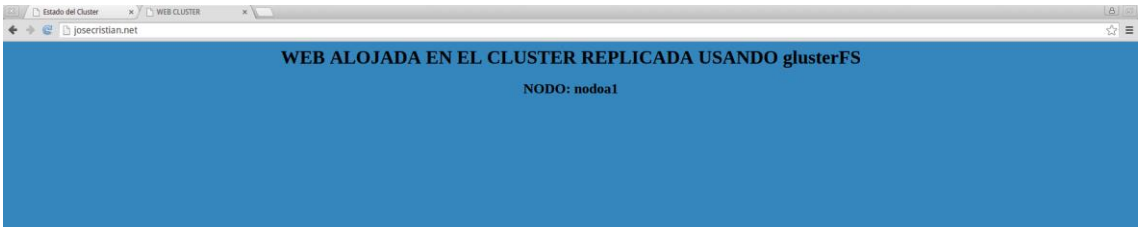


***Imagen 11.3. Web.***

Como vemos el nodo que nos esta ofreciendo la web es el nodo **nodoa1** del cluster A que es el nodo activo activo en ese momento, si volvemos a recargar la web veremos que el nodo que nos ofrece la web es el nodo activo del Cluster B.

El balanceo entre los clusters se realiza usando el balanceador de carga activo en el momento entre LB1 y LB2, ya que al estar replicados aun habiendo un fallo en uno de los balanceadores la web seguirá funcionando.

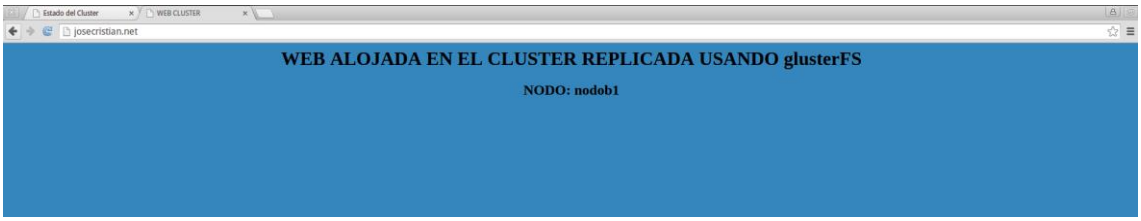


***Imagen 11.4. Web.***

NOTA: El contenido de la web esta replicado ya que el nombre del nodo es una variable PHP que extrae el nombre del servidor, lo que permite distinguir el nodo que ofrece la web en el entorno de pruebas.

Si apagamos un nodo (simulamos un fallo) veremos que el nodo que responde es otro nodo del cluster que se situa como **activo**, apageremos el nodo **nodoa1**.

Si accedemos a la web veremos que ahora nos reponde el nodo **nodoa2** del cluster A y el mismo nodo del cluster B ya que en ese cluster no hay error.

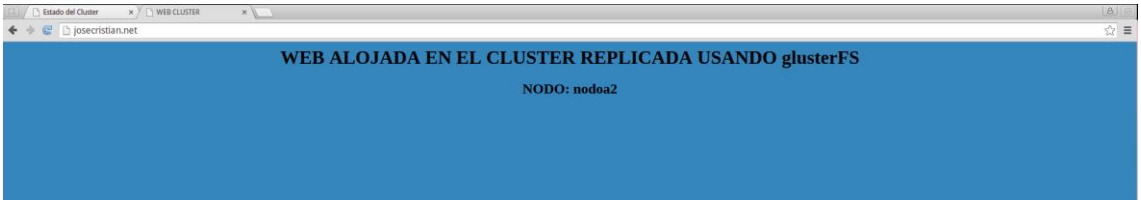


***Imagen 11.5. Web.***

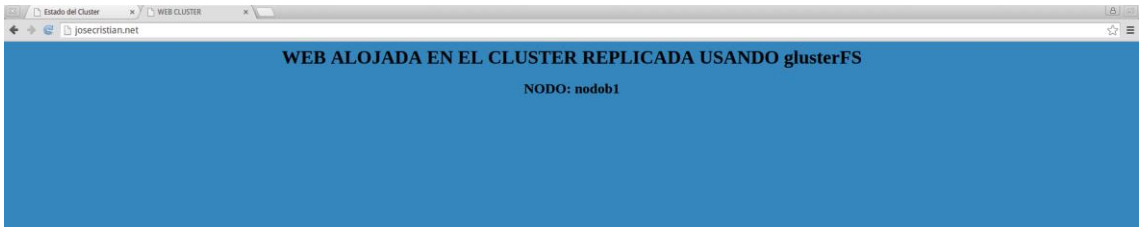


***Imagen 11.6. Web.***

Si accedemos al panel de monitorización del cluster veremos que se muestra el nodo fallido **nodoa1** y el nodo al que se han pasado los recursos **nodoa2**.

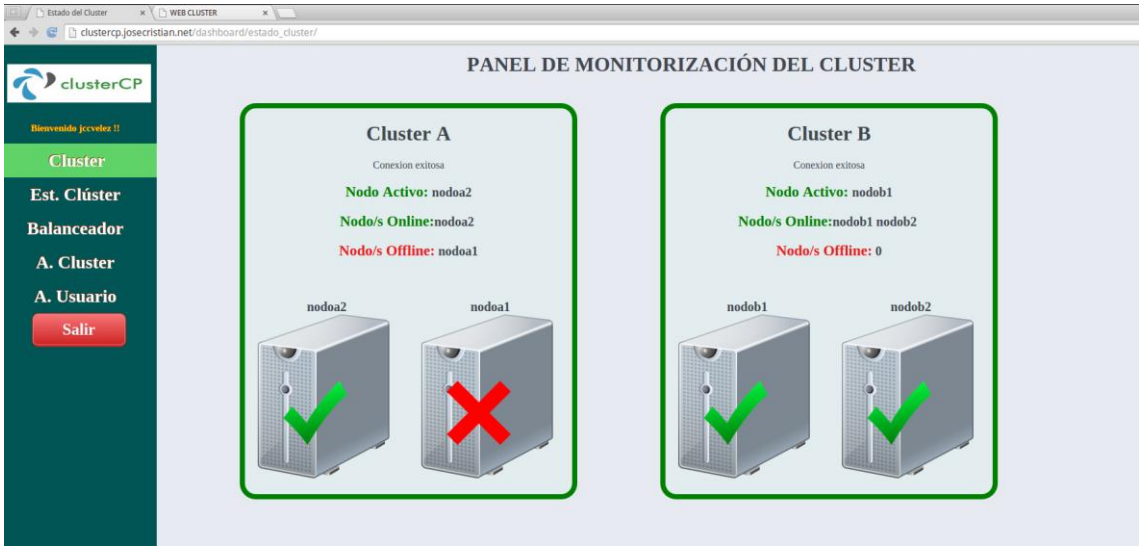


***Imagen 11.7. Nodo fallido.***

En el cluster activo podemos comprobar los nodos erróneos usando el comando **pcs status nodes**.

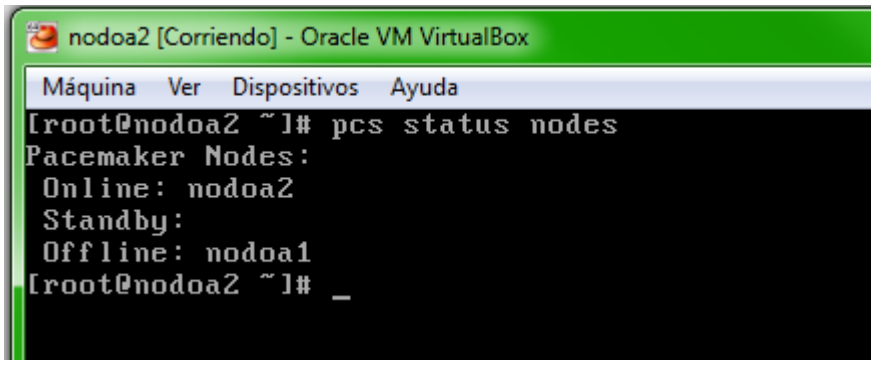


***Imagen 11.8. Estado de los nodos del CLUSTER A.***

Si accedemos al log **/var/log/cluster/error_cluster.log** veremos el nodo que presenta el error.

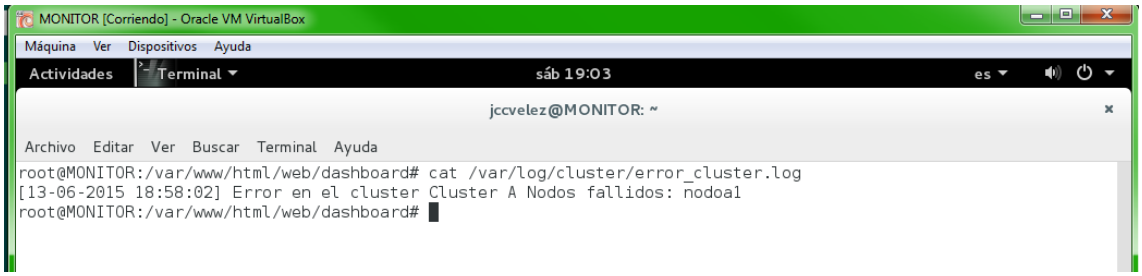


***Imagen 11.9. Log de error.***

Tambien podemos ver el log **/var/log/cluster/ok_cluster.log** que informa que la notificación del fallo se ha enviado correctamente por correo electrónico.

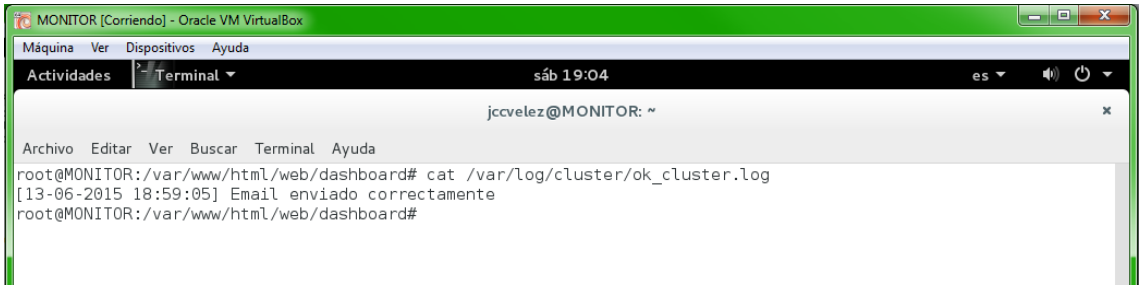


***Imagen 11.10. Log OK, email enviado correctamente.***

Veremos el correo recibido.


***Imagen 11.11. Corre recibido.***

Ahora accederemos a la página de **estadísticas del cluster** http://clustercp.josecristian.net/dashboard/estado_cluster y comprobaremos las caidas del nodo.

Veremos las caídas por fecha, caídas por nodo y memoria RAM usada por cada cluster (nodo activo).

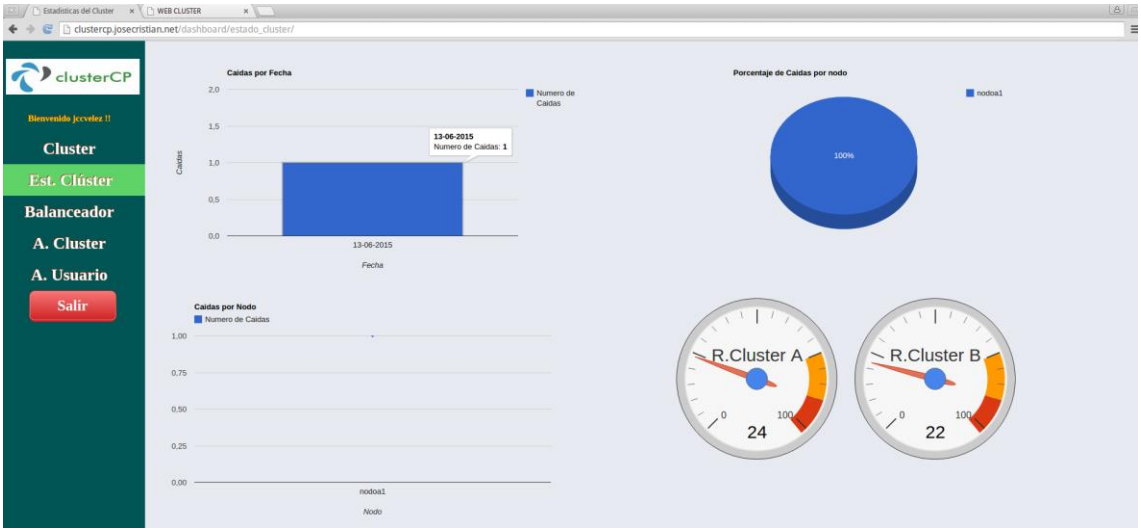


***Imagen 11.12. Estadisticas de los nodos y RAM usada por cada cúster.***

Para comprobar la alta disponibilidad de los balanceadores de carga apagaremos uno de ellos simulando un fallo y veremos como el usuario sigue navegando correctamente.

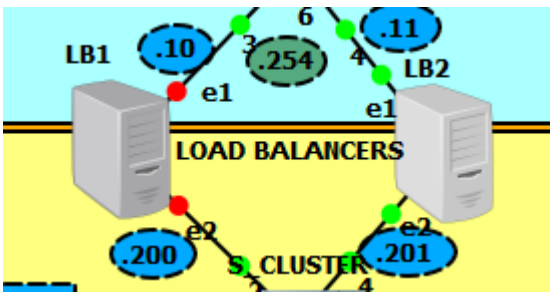


***Imagen 11.13. Estructura de los balanceadores.***

Accedemos a la web.


***Imagen 11.14. Web.***

Podremos comprobar el ping a la dirección IP virtual de los balanceadores de carga (KeepAlived).

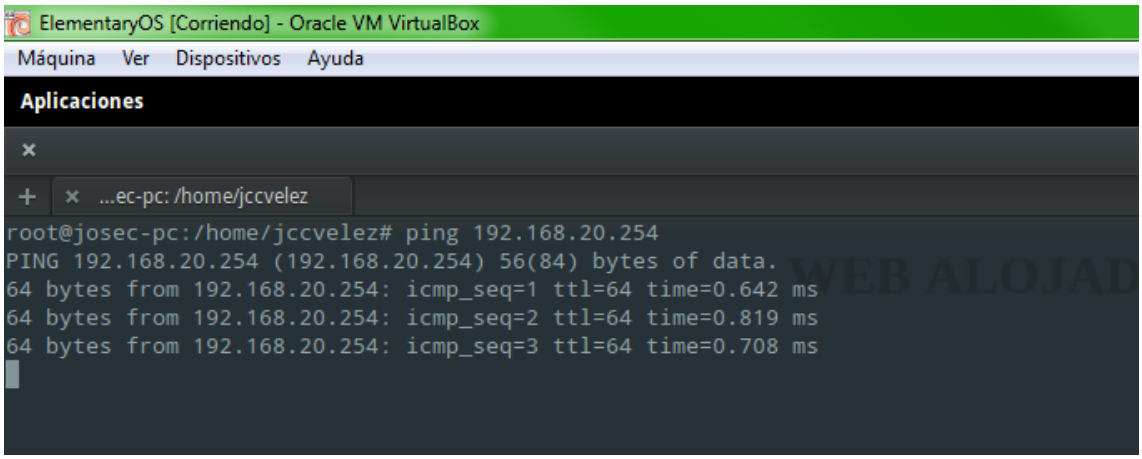


***Imagen 11.15. Ping correcto al balanceador con un servidor fallido.***

Ahora apageremos todos los nodos de cluster **CLUSTER B** para comprobar que con un un único servidor (**nodoa2**) seguiremos teniendo acceso a la web.

Comprobaremos en el panel de monitorización del cluster que están caidos todos los nodos del **CLUSTER B**.

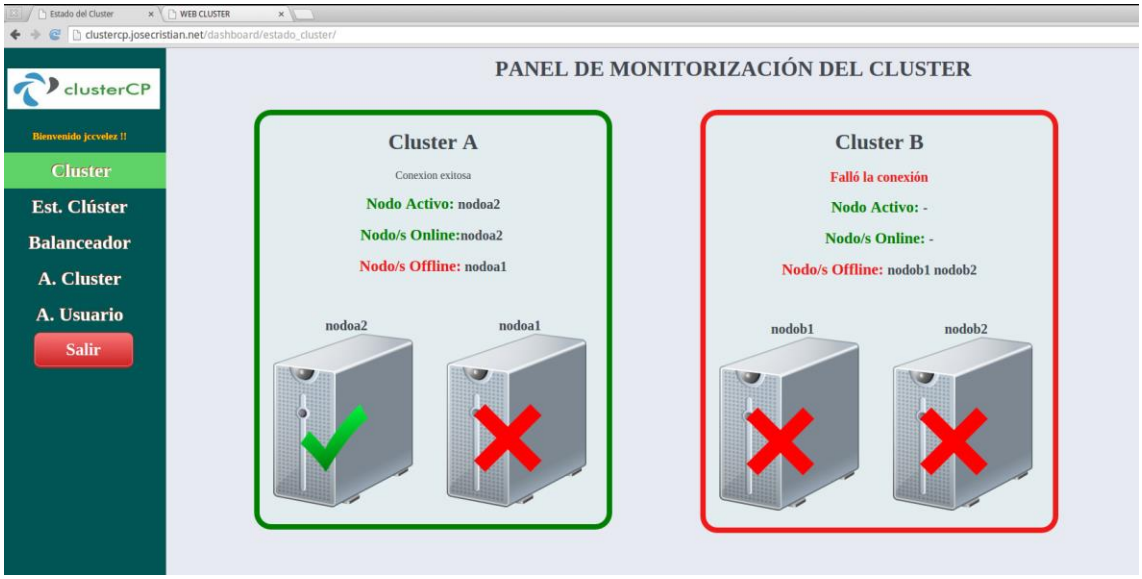


***Imagen 11.16. Clúster completo fallido.***

Ahora acceremos a la web y veremos que si recargamos únicamente nos muestra la web el nodo **nodoa2**.


***Imagen 11.17. Web.***

Si accedemos al panel de estadísticas del balanceador pulsando el **botón de estadísticas dentro del menú** veremos que el BACKEND del **CLUSTER B** esta caído y el balancedor no reenviara las peticiones a ese cluster.


***Imagen 11.18. Acceso a las estadísticas del balanceador de carga.***

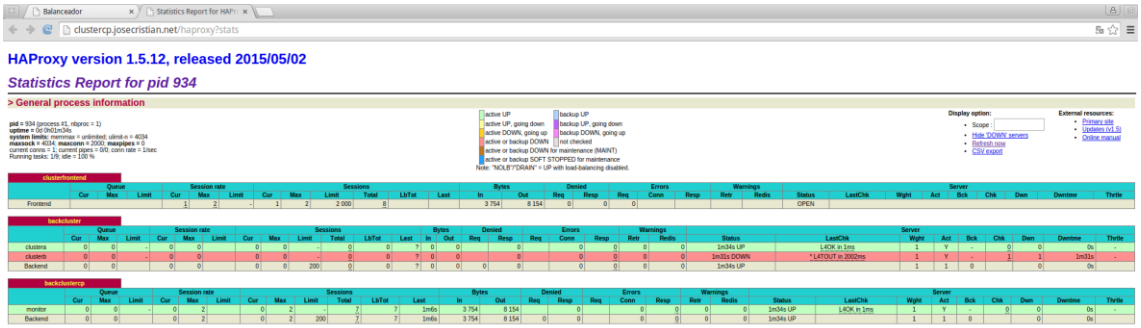


***Imagen 11.19. Panel de estadísticas del balanceador de carga.***

Tambien podremos ver el correo del cluster fallido ya que no se puede hacer ping.


***Imagen 11.20. Notificacion por correo de clúster completo fallido.***

Una vez solucionado el fallos (todo esta correctamente funcionando) comprobaremos el cambio del algoritmo en los servidores.

Antes de cambiar el algoritmo veremos que cada conexión que realizamos a la pagina web se cambia de cluster ya que el algoritmo es RoundRobin, es decir, las peticiones que llegan se distribuyen entre cada cluster.

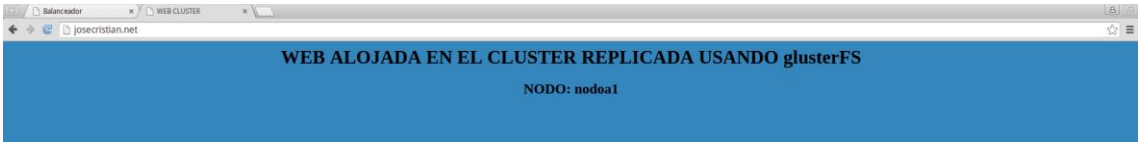


***Imagen 11.21. Web.***

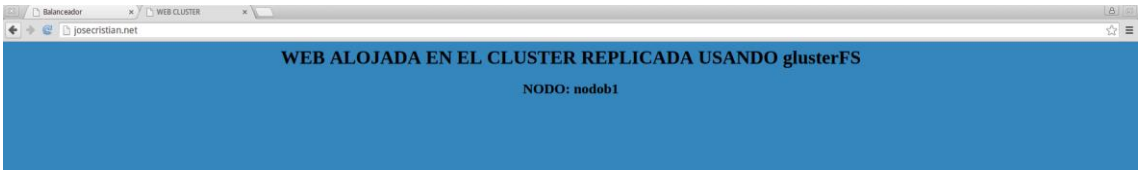


***Imagen 11.22. Web.***

Ahora cambiaremos el algoritmo a través del formulario del **balanceador** elegiremos el algoritmo **source** cuya funcion el enviar la conexión de la misma dirección IP al mismo servidor lo que permite que un mismo usuario navegue siempre en el mismo servidor.


***Imagen 11.23. Cambio del algoritmo de balanceo de carga.***

Ahora navegaremos a la web **http://josecristian.net** y veremos que en las futuras recargas navegamos en el mismo nodo.


***Imagen 11.24. Web.***


***Imagen 11.25. Web.***

Si añadimos un nuevo cluster a la infraestructura usaremos el formulario situado en el menú **A. Cluster** en el que introduciremos el nombre, ip y nodos (separados por un espacio).

Supongamos que queremos añadir el **CLUSTER C** con la dirección IP **192.168.50.252** y los nodos **nodoc1 nodoc2** y **nodoc3**.

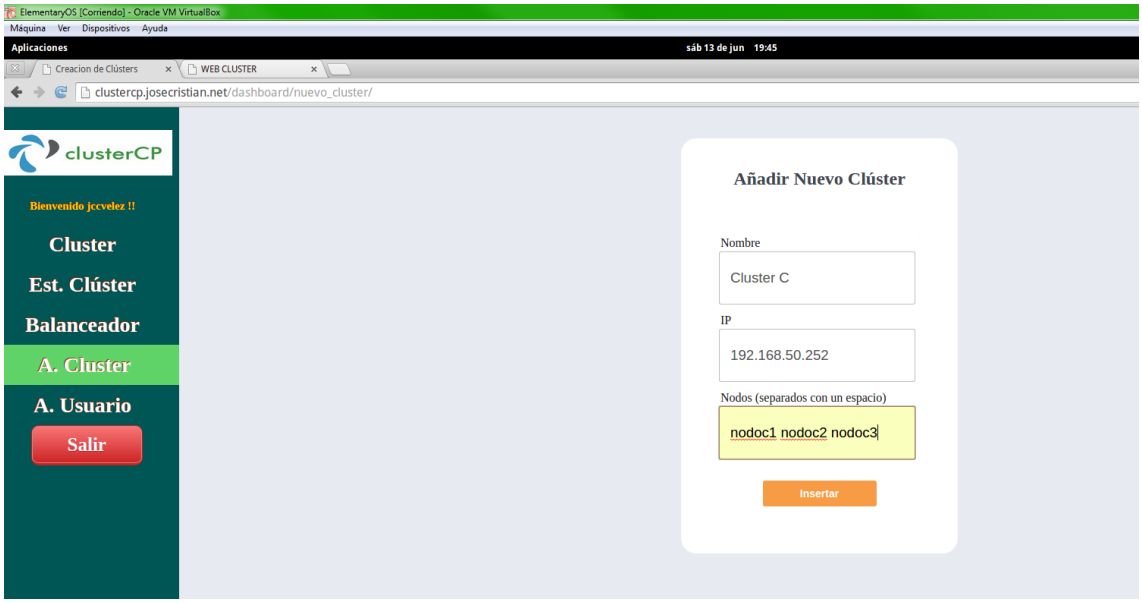


***Imagen 11.26. Formulario para añadir un nuevo clúster.***

Una vez añadido veremos el mensaje de que se han añadido correctamente.

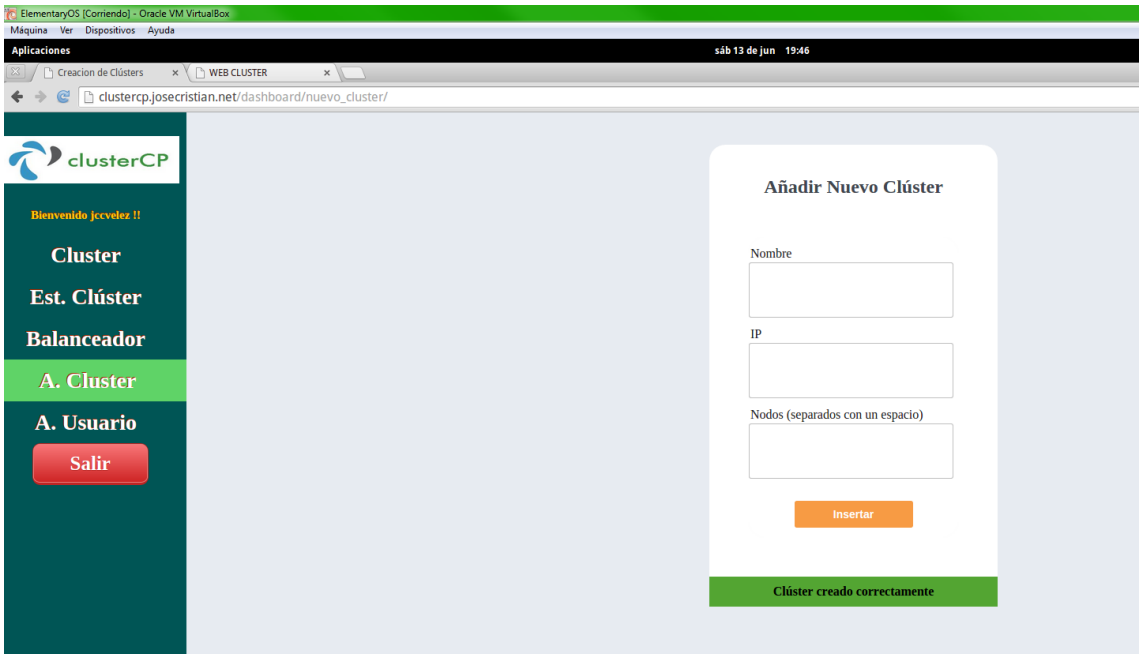


***Imagen 11.27. Mensaje de clúster añadido correctamente.***

Si ahora accedemos al panel de monitorización del cluster veremos el nuevo cluster pero con el error de que no se puede conectar (ya que no existe). Veremos los nodos que los componen.

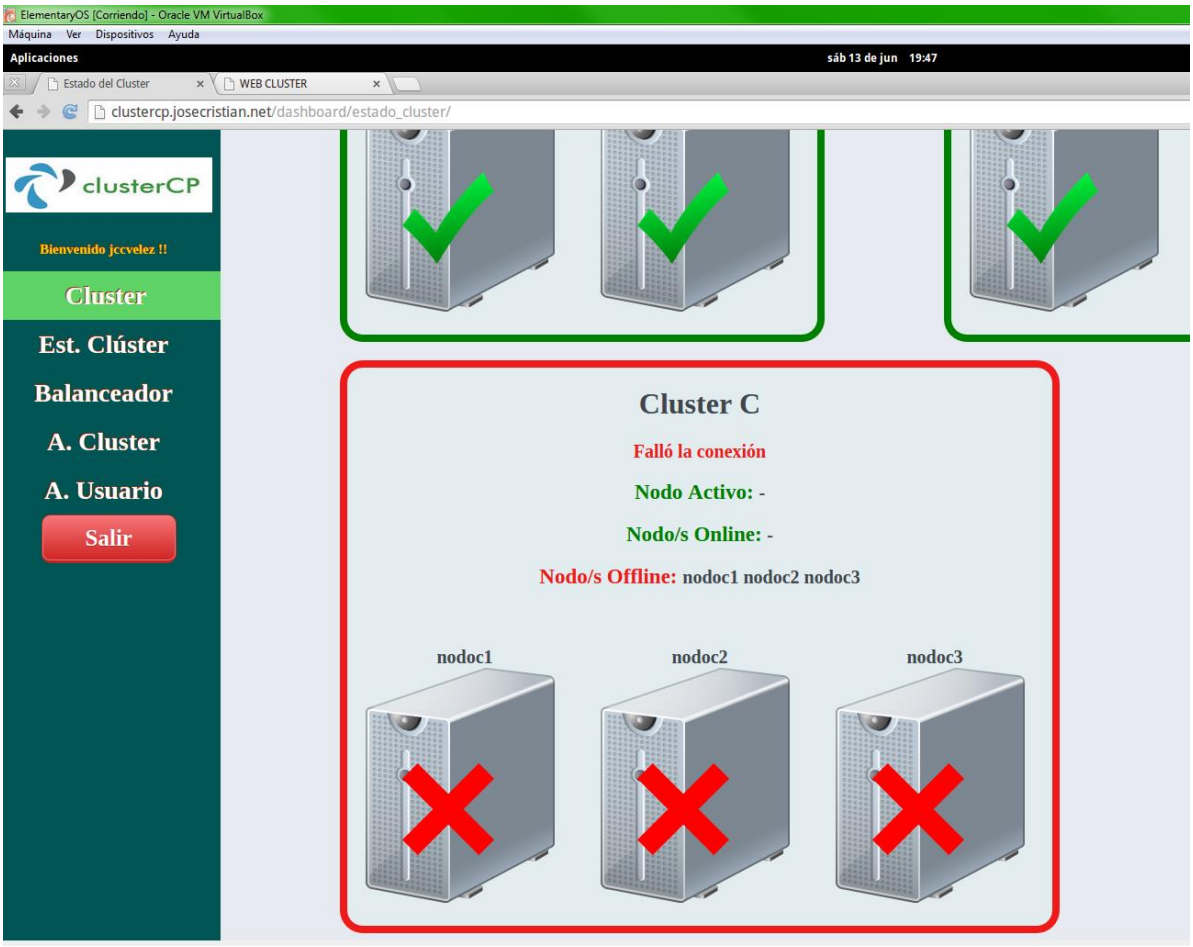


***Imagen 11.28. Estado del clúster añadido.***

# 12. Mejoras en versiones posteriores

Dado que esta en una primera versión del **clusterCP** se pueden ir realizando mejoras y añadiendo nuevas funcionalidades en versiones posteriores como las siguientes:

- Permitir añadir Balanceadores de carga a través de un formulario
- Monitorizacion del balanceador de carga si hay un fallo.
- Cambiar nodos del cluster
- Realizar la instalación de un nuevo cluster a través del panel

# 13. Conclusiones

Tras la finalización de este proyecto he sacado buenas conclusiones acerca de la implantación de una infraestructura de alta disponibilidad, ya que es muy importante planificar los detalles para el posterior despliegue, ya que as'i evitaremos problemas que podían haberse corregido al principio y ahorra tiempo y dinero. Debido a la gran cantidad de software libre que existe para la implantación de clusters, proxys, etc. no es necesario invertir dinero en software teniendo el inconveniente del soporte, ya que en el software de pago normalmente se ofrece un soporte oficial ya sea de pago o no, sin embargo en el software libre mucho software se rige y evoluciona a través de la comunidad de usuarios y hay pocas empresas que ofrezcan soporte oficial real.

# 14. Referencias


**Wiki Alta disponibilidad - corosync**

- http://clusterlabs.org/wiki/Initial_Configuration

**Pacemaker**

- http://www.linux-ha.org/wiki/Pacemaker

**PCS**

- http://clusterlabs.org/doc/fr/Pacemaker/1.1-pcs/html/Clusters_from_Scratch/_configuring_corosync.html

**GlusterFS**

- http://gluster.readthedocs.org/en/latest/

**PHP**

- http://php.net/docs.php

**MongoDB**

- http://docs.mongodb.org/manual/

**Google Charts**

- https://developers.google.com/chart/